Web Application Security:

Input Validation with JS and PHP

By Dave Bernardy

Overview

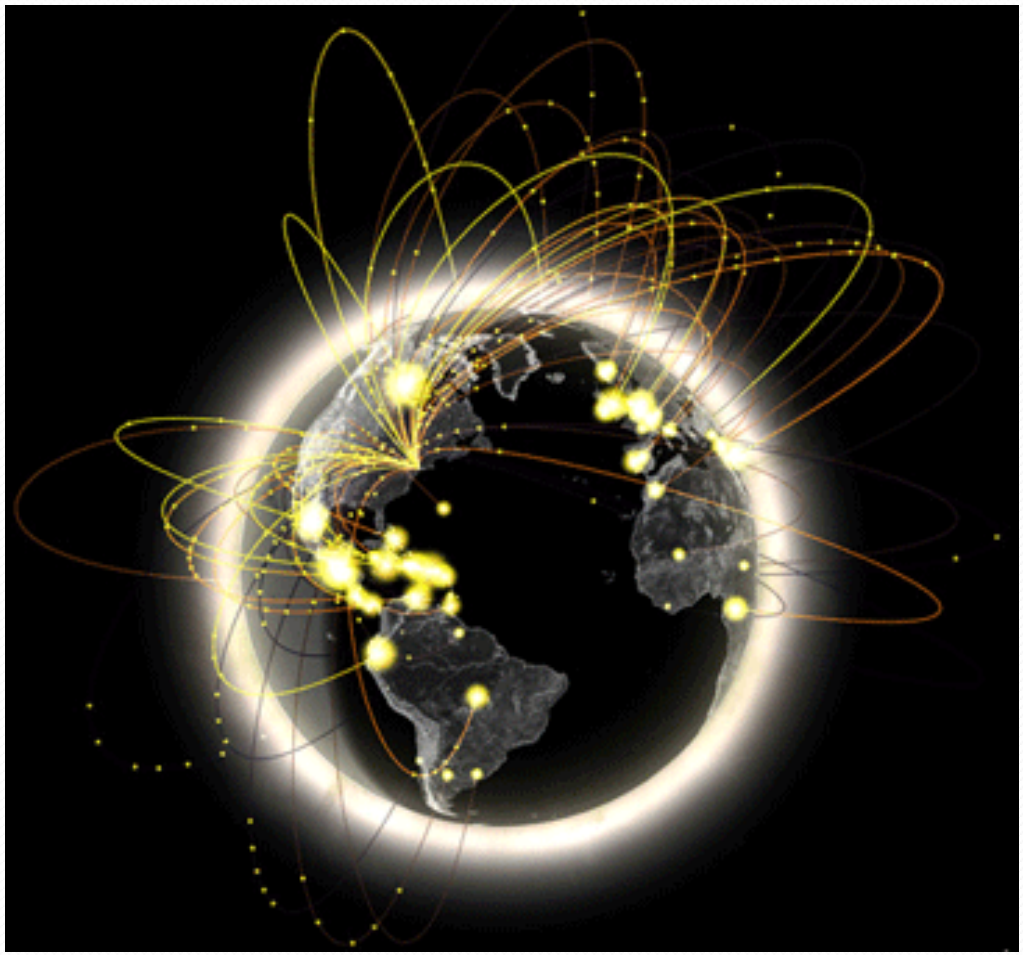
- Motivation
- Background Information
  - Client – Server Model
  - PHP
- HTML Injections
  - SQL
  - XSS
- Doubling Security
  - Validate ALL Input
    - Client-Side Validation (JS)
    - Server-Side Validation (PHP)
- Other Considerations

Motivation

- Awareness

- Valuable data

Motivation



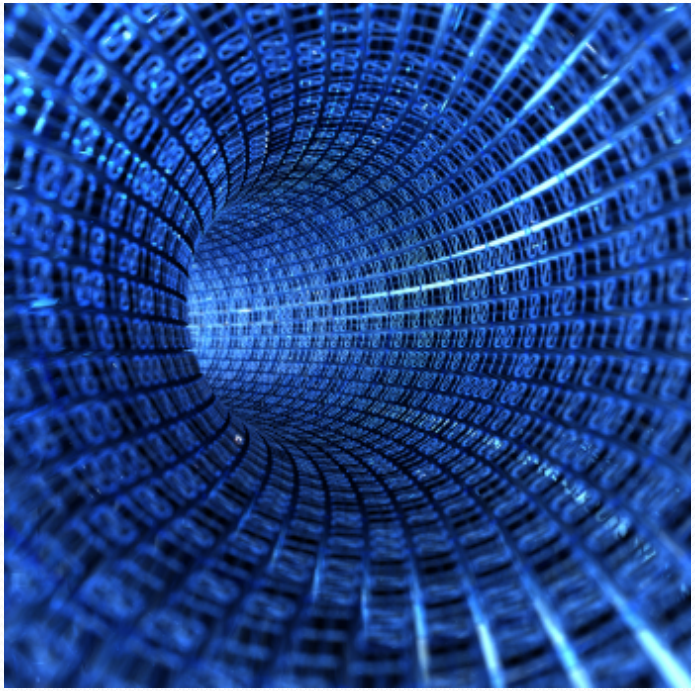
Part of Everyday Life

Motivation

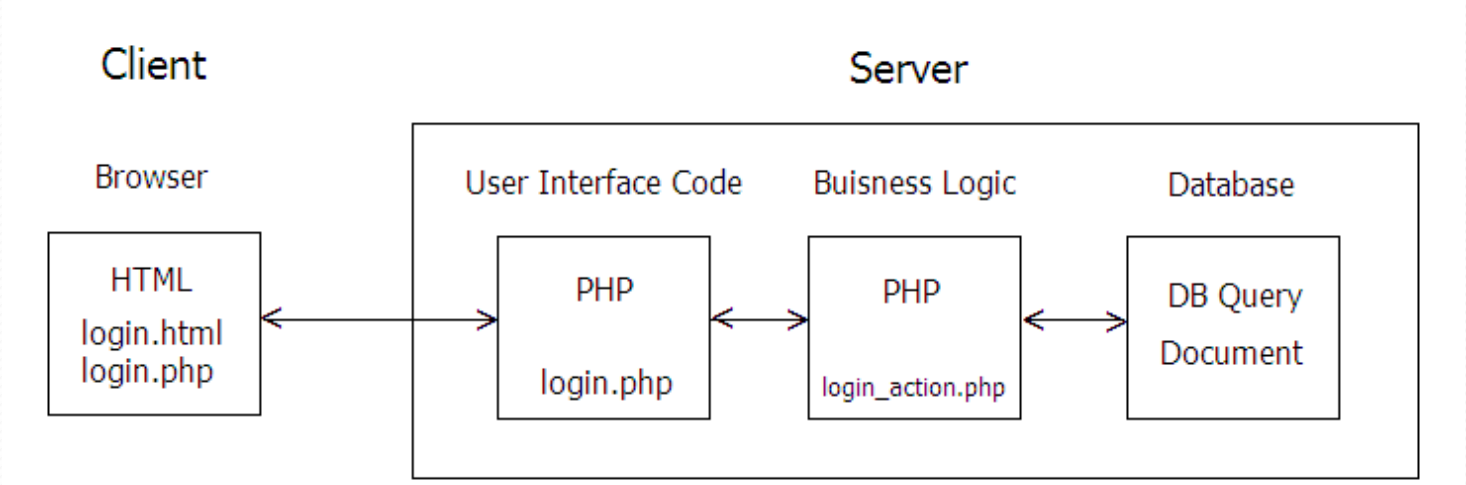
Hackers

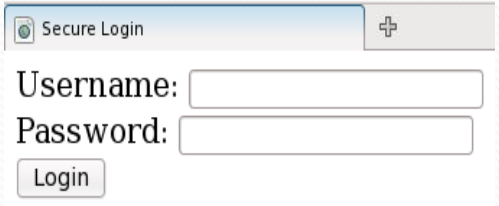
Cyberspace and the Law

- Computer Fraud and Abuse Act (1986)
- Laws Against:
  - Credit Card Abuse
  - Stolen Property
  - Viruses & Worms
  - Trespassing
  - Manipulating Data



Client – Server Model



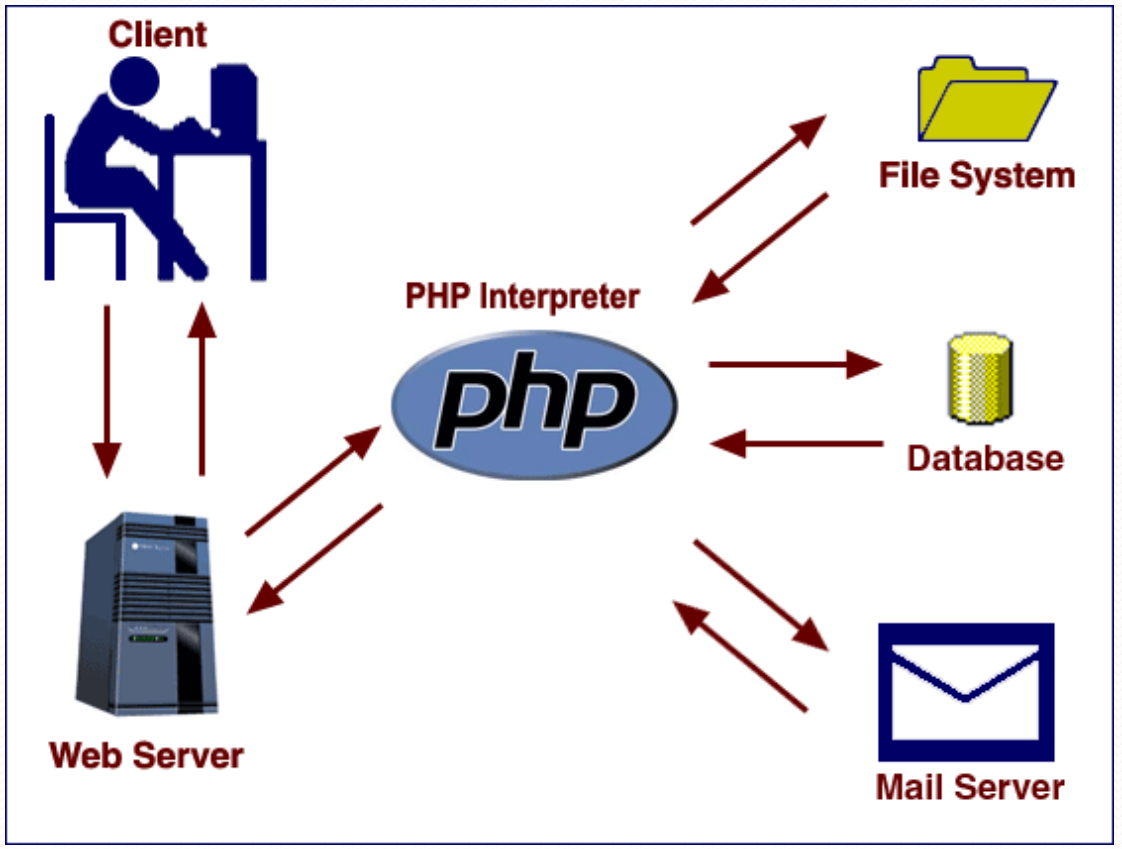



```php
<?php

// Code goes here.

?>
```

```
File  Edit  View  Terminal  Help
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM client;
+------+----------+----------+---------+----------+-------------------
| u_id | u_name   | p_word   | f_name  | l_name   | email
+------+----------+----------+---------+----------+-------------------
|    1 | berndav  | basketball | Dave  | Bernardy | djbernardy@csbsju.
|    2 | monkeyman | banana  | Dustin  | Johnson  | monkeyman@gmail.co
|    3 | mjordan  | airman   | Michael | Jordan   | airJordan@gmial.co
+------+----------+----------+---------+----------+-------------------
3 rows in set (0.00 sec)

mysql> SELECT * FROM entry;
+------+----------+-------------------------------------------------+-
| e_id | u_name   | msg                                             | t
+------+----------+-------------------------------------------------+-
|    1 | berndav  | I will miss this place!!!                       | 2
|    2 | berndav  | I think I missed mass this past weekend.. uhh oh | 2
|    3 | monkeyman | Reminds me of heaven                           | 2
|    4 | mjordan  | I believe I can fly                             | 2
+------+----------+-------------------------------------------------+-
```

PHP: Hypertext Preprocessor

- Server-side scripting language
- Dynamic pages
- Functions
- Validation
- Query databases
- 

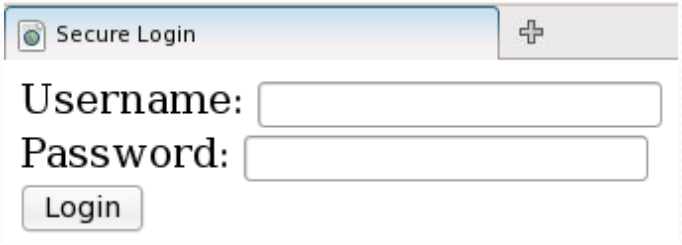


PHP: Hypertext Preprocessor

- PHP embedded in HTML

```
1  <html>
2  <head>
3  </head>
4          <body>
5                  <?php
6                          $txt="Hello World";
7                          echo $txt;
8                  ?>
9          </body>
10 </html>
```

HTML



Client-Side

```
<form name="login" action="insecure_login_action.php" method="post">
        <label>Username:</label> <input name="u_name" /><br>
        <label>Password:</label> <input name="p_word" type="password" /><br>
        <input value="Login" type="submit" />
</form>
```

Server-Side

```
127 $username = $_POST['u_name'];
128 $password = $_POST['p_word'];
```

HTML Injections

- Failure to validate input
  - SQL Injections
  - Cross-site scripting (XSS)

NEVER
Trust User Input

Structured Query Language (SQL)

- Language designed for managing data in a relational database management system

Query 

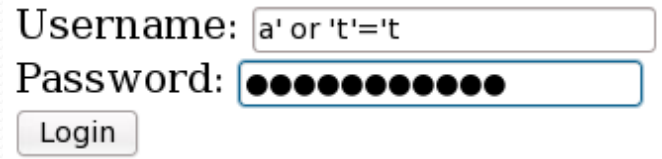
```
SELECT   title
FROM   Books
WHERE   price > 100.00
```

Results 

```
Title
-----------------------------------
The Very Hungry Caterpillar
Where the Sidewalk Ends
The Cat in the Hat
```

SQL Injections

Client-Side



Server-Side

```
127 $username = $_POST['u_name'];
128 $password = $_POST['p_word'];
```

=

```
133 $username = "a' OR 't'='t";
134 $password = "a' OR 't'='t";
```

SQL Injections

Client-Side

Username: a' or 't'='t

Password: ●●●●●●●●●●

Login

```
133
134 $queryString = "SELECT * FROM client WHERE u_name=$username AND p_word=$password",
135
```

=

```
220
221 $queryString = "SELECT * FROM client WHERE u_name='a' OR 't'='t' AND p_word='a' OR 't'='t';"
222
```

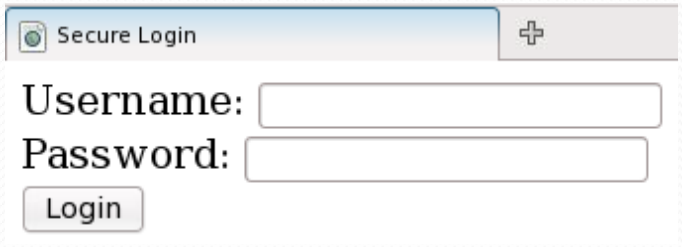
Demonstration

- Malicious Input
  - Gain access to web application
  - http://devsrv.cs.csbsju.edu/~djbernar/security/login.php

Defense: Client-Side

- Fast
- HTML
  - Input box maxLength
- JavaScript
  - Type
  - Length
  - Regular Expression

Defense: Client-Side




```
<form name="login" action="secure_login_action.php" onsubmit="return (validateString(this.u_name, 'Please enter a
        valid username', 3, 25) && validateString(this.p_word, 'Please enter a password', 3, 25));" method="post">
        <label>Username:</label> <input name="u_name" maxLength="25" /><br>
        <label>Password:</label> <input name="p_word" type="password" /><br>
        <input value="Login" type="submit" />
</form>
```

Defense: Client-Side (JS)

```
<script type="text/javascript" language="javascript">

        function validateString(field, msg, min, max) {
                if (!field.value || field.value.length < min || field.value.length > max) {
                        alert(msg);
                        field.focus();
                        field.select();
                        return false;
                }
                else{
                        var patt1=new RegExp("[^a-z]");
                        if(patt1.test(field.value)!=0){
                                alert("Failure. Your string had bad chars!!!");
                                return false;
                        }
                }
                return true;
        }
</script>
```

Defense: Client-Side (JS)

- Easy to bypass
  - Create new HTML document
  - Disable scripting in browser

```
1 <html>
2         <body>
3                 <form name="login" action="secure_login_action.php" method="post">
4                         <label>Username:</label> <input name="u_name" /><br>
5                         <label>Password:</label> <input name="p_word" type="password" /><br>
6                         <input value="Login" type="submit" />
7                 </form>
8         </body>
9 </html>
```

Defense: Server-Side (php)

- PHP
  - Ensure field has been filled out

```
16 isset($_POST['u_name'];
```

  - Data Type (String, Integer, Boolean, etc.)

```
21 isString($_POST['u_name'];
22 isInteger($_POST['u_name']);
```

Defense: Server-Side (php)

- PHP
  - Length

```
18 strlen($_POST['u_name'])
```

  - Regular Expression

```
45 // regEx Test!!!!
46 function check_alnum_regex($var){
47   return preg_match('/[^a-zA-Z0-9]/', $var) ? TRUE : FALSE;
48 }
```

```
16 ctype_alnum($_POST['u_name']);
```

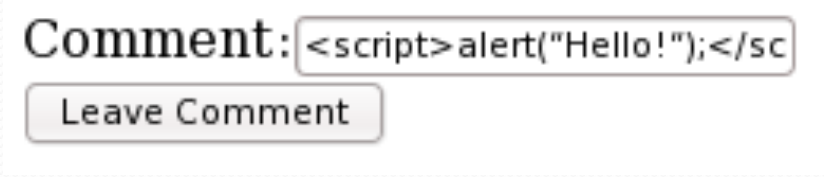
```php
function sanityCheck($string, $type, $length){
        // Assign the type
        $type = 'is_'.$type;

        if(!$type($string)){
                return FALSE;
        }

        // Is the input empty
        elseif(empty($string)){
                return FALSE;
        }

        // Check string length
        elseif(strlen($string) > $length){
                return FALSE;
        }

        else{
                return TRUE;
        }
}
```

```php
//Check for String, and length of 25
if(isset($_POST['u_name']) && (sanityCheck($_POST['u_name'], 'string', 25) != FALSE)){
        // username and password are a-zA-Z0-9
        if((int) ctype_alnum($_POST['u_name']) && (int) ctype_alnum($_POST['p_word'])){
                $username = $_POST['u_name'];
                $password = $_POST['p_word'];
        }else{
                die ("Checkpoint 3 FAILED - Username must be a-zA-Z0-9!!!  FIX THEM");
        }
}
else{
        die("Checkpoint 1 FAILED - Please enter a VALID username");
}
```

Demonstration

- Test Validation
  - http://devsrv.cs.csbsju.edu/~djbernar/security/login.php

XSS

- User Trust
- JavaScript
- Example
  - <script>alert(“Hello!“);</script>
  - 

Demonstration

- Malicious Input
  - Steal Cookies
  - http://devsrv.cs.csbsju.edu/~djbernar/security/login.php

Not that Easy

-
- <script src=http://ha.ckers.org/xss.js></script>
-
- <script>window.open('http://www.evil.site.com', 'width=400, height=300');<script>

Defense: Server-Side (php)

```
$comment = $_POST["comment"];
```

```
$comment = strip_tags($comment);
```

alert(“Hello World”);

```
$comment = htmlspecialchars($comment);
```

<script>alert(“Hello World”);</script>

<script>alert("Hello!!!");</script>

Not that Easy

alert(“hello world”)

<script>alert(“hello world”);<script>

Demonstration

- Comment Sanitization
  - http://devsrv.cs.csbsju.edu/~djbernar/security/login.php

Other Considerations

- Integrate Security from the beginning
- Output validation
- php Preferred Statements
- SQL Stored Procedures
- Encryption
- SSL
- Other Attack Vectors
  - CSRF
  - AJAX
  - Mobile Phone Access

Review

- Importance of Validation
- Background Information
  - Client – Server Model
- HTML Injections
  - SQL
  - XSS
- Doubling Security
  - Validate ALL Input
  - Client-Side Validation (JS)
  - Server-Side Validation (PHP)
- Other Considerations

References

- http://php.net/index.php/
- http://www.w3schools.com/php/
- http://www.phpro.org/
- http://www.webreference.com/programming/
- http://ha.ckers.org/xss.html