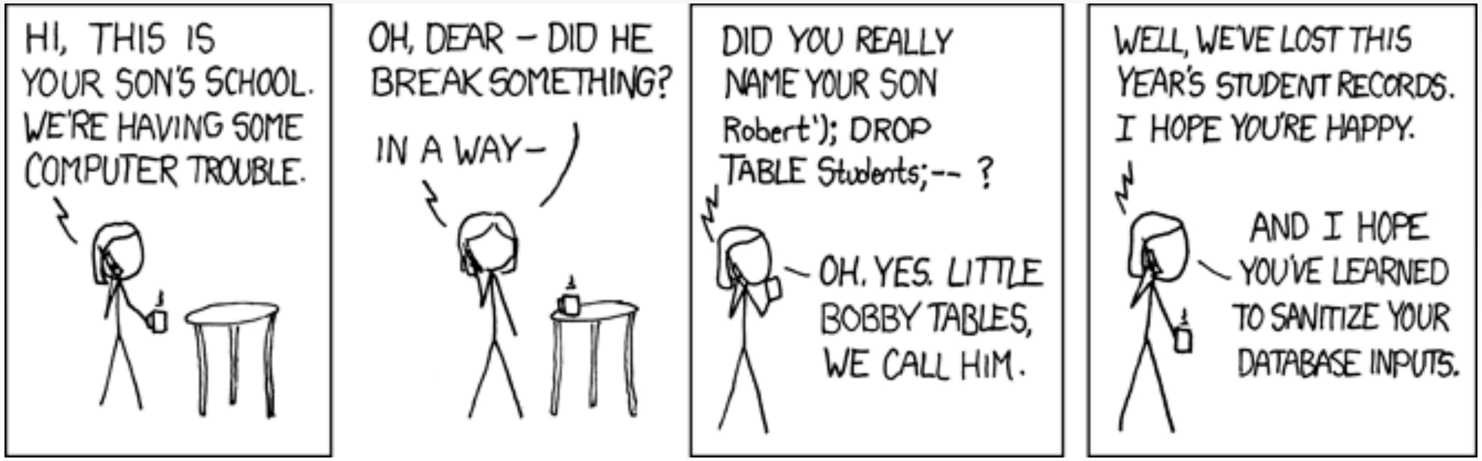
SQL Injections

Michael Hill, CISSP

June 14, 2013

Definition

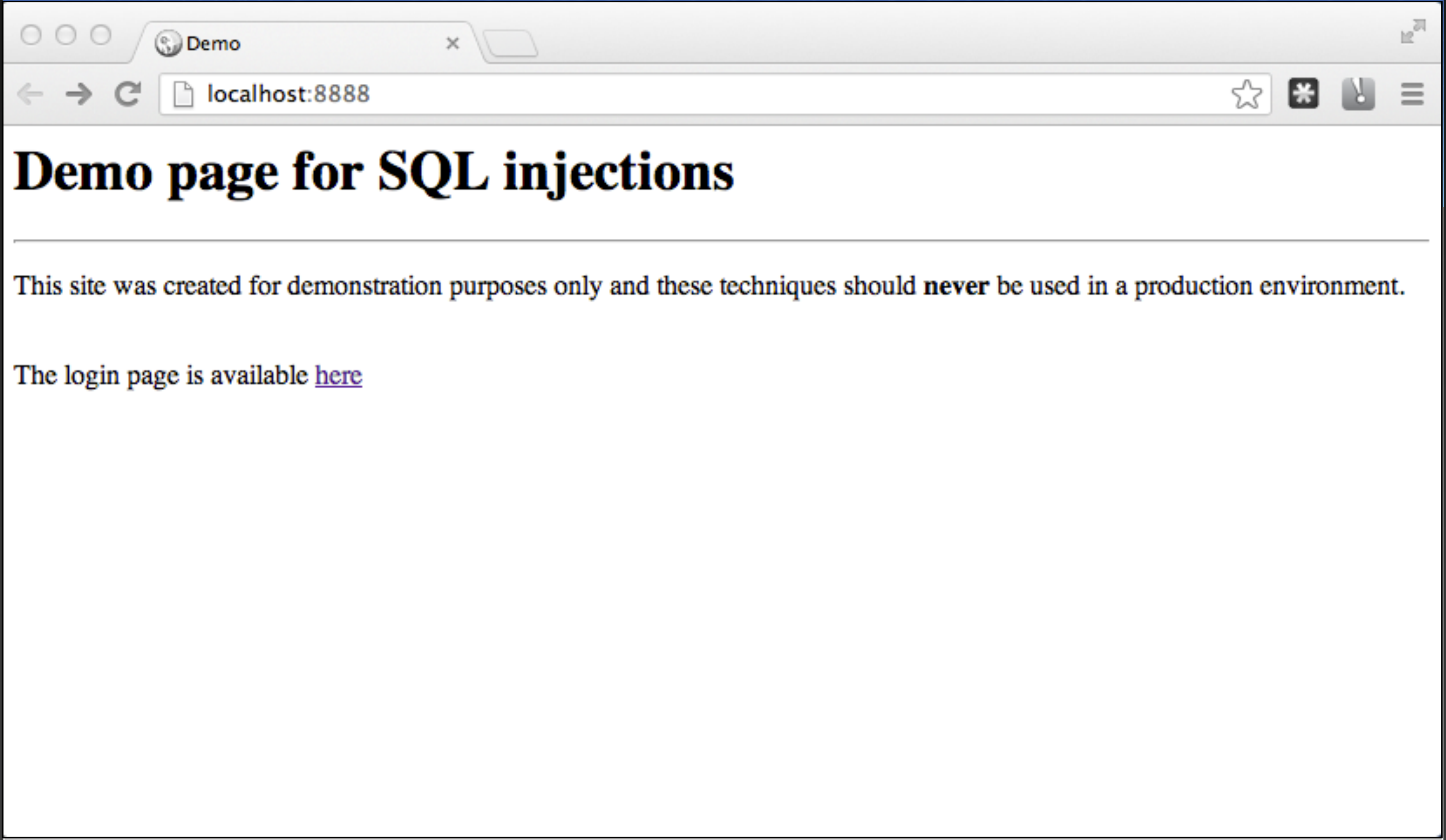
- An SQL injection is a computer attack in which malicious code is embedded in a poorly-designed application and then passed to the backend database. The malicious data then produces database query results or actions that should never have been executed.


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

How real is the threat?

- Injection flaws have consistently been in the OWASP top 10 list.
  - 2004: A6 Injection Flaws
  - 2007: A2 Injection Flaws
  - 2010: A1 Injection
  - 2013: A1 Injection

Demo Page

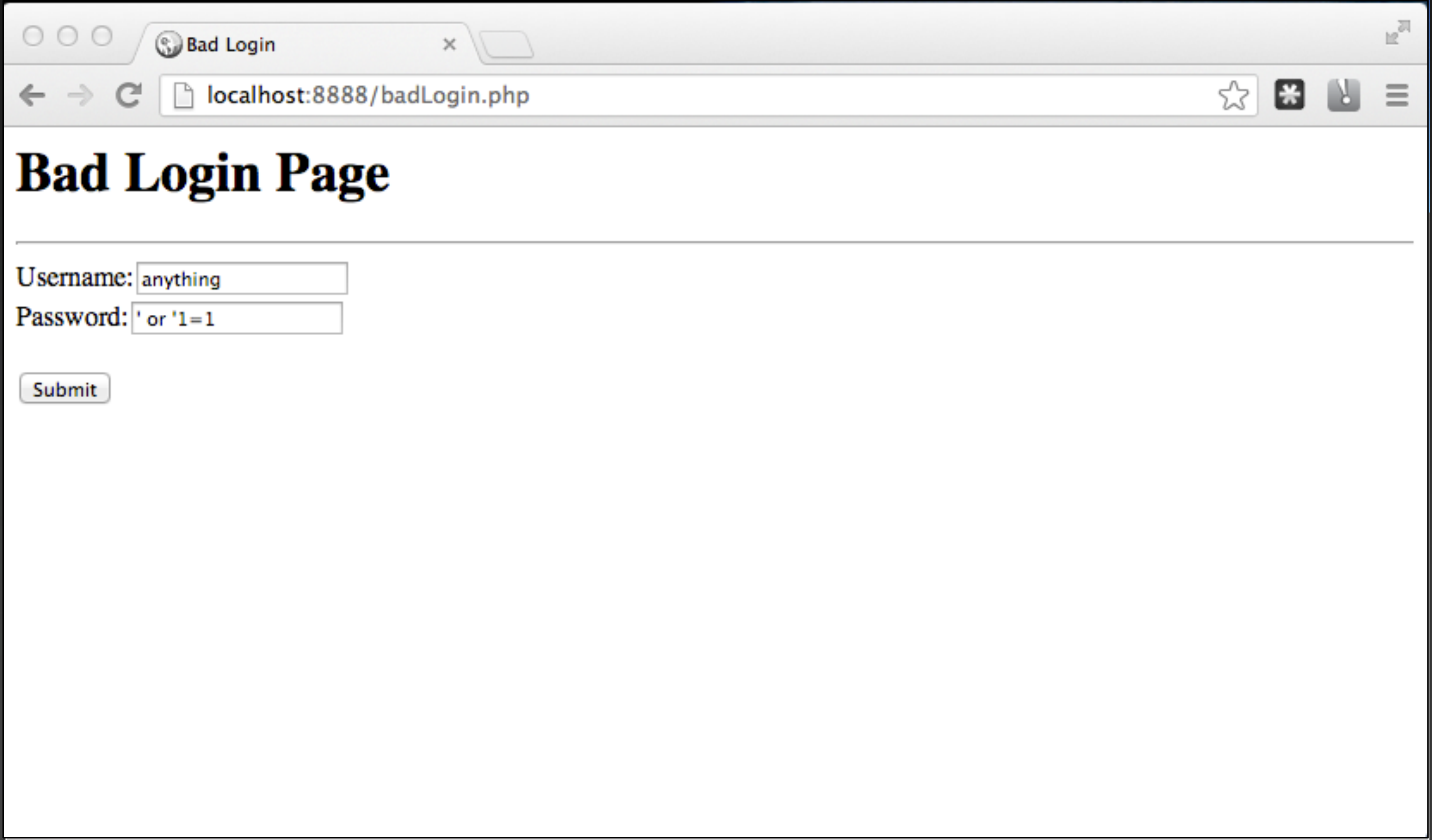


Demo page for SQL injections

This site was created for demonstration purposes only and these techniques should **never** be used in a production environment.

The login page is available here

Vulnerable Login Page

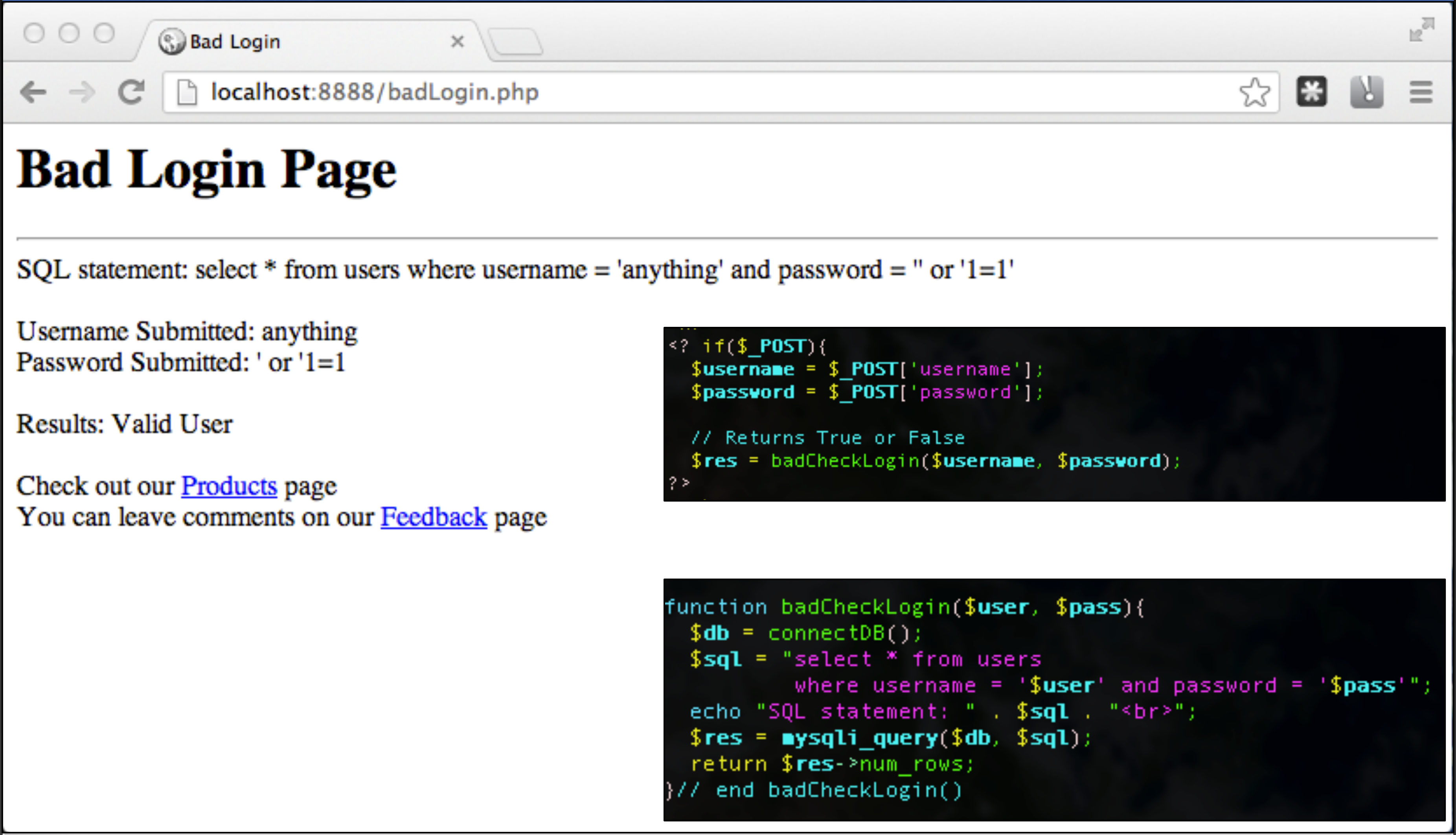


Bad Login Page

Username: anything

Password: ' or '1=1

SQL Injection - Login



Bad Login

localhost:8888/badLogin.php

Bad Login Page

SQL statement: select * from users where username = 'anything' and password = '' or '1=1'

Username Submitted: anything
Password Submitted: ' or '1=1

Results: Valid User

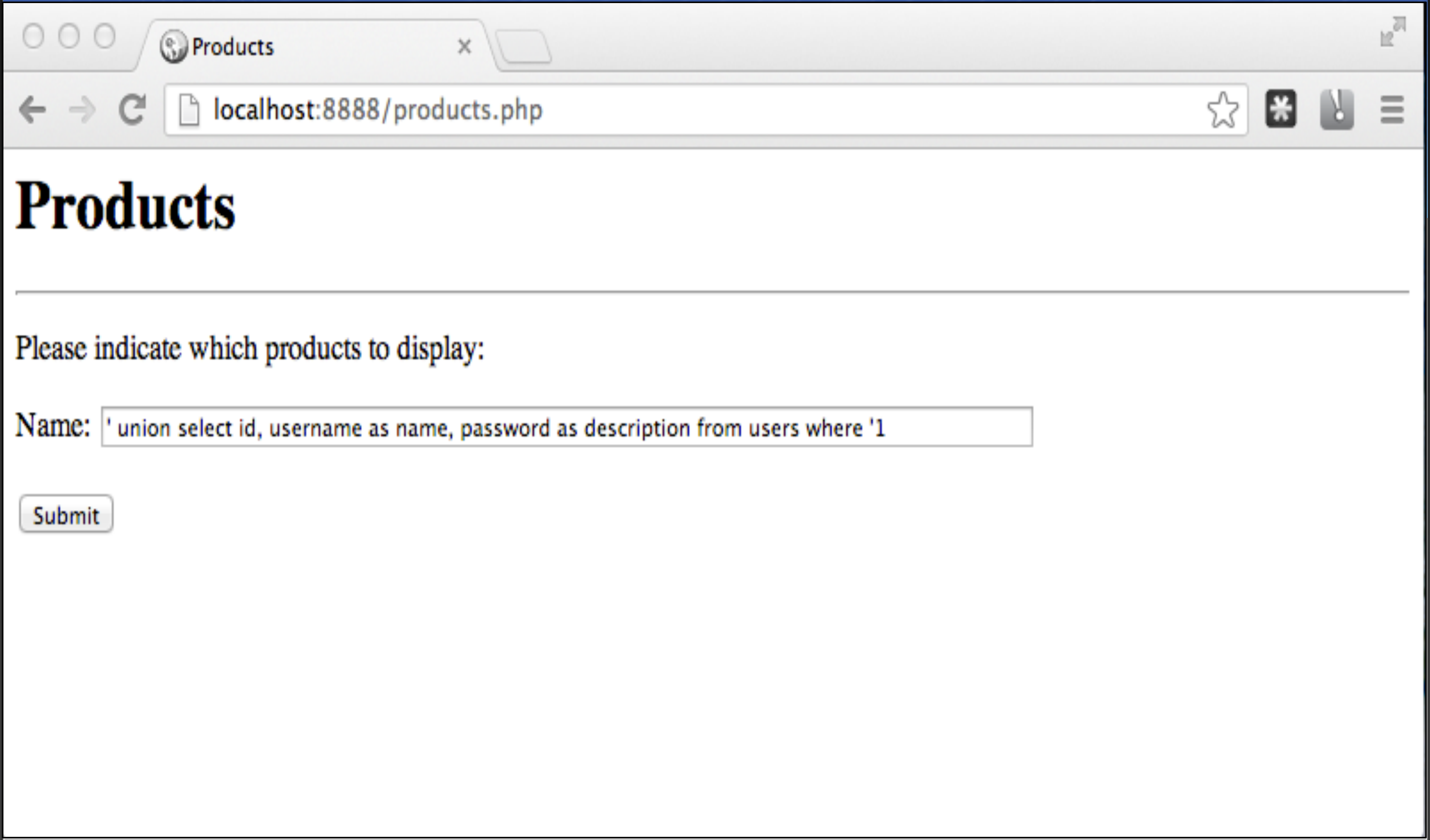
Check out our Products page
You can leave comments on our Feedback page

```
<? if($_POST){
  $username = $_POST['username'];
  $password = $_POST['password'];

  // Returns True or False
  $res = badCheckLogin($username, $password);
?>
```

```
function badCheckLogin($user, $pass){
  $db = connectDB();
  $sql = "select * from users
          where username = '$user' and password = '$pass'";
  echo "SQL statement: " . $sql . "<br>";
  $res = mysqli_query($db, $sql);
  return $res->num_rows;
}// end badCheckLogin()
```

Products Page

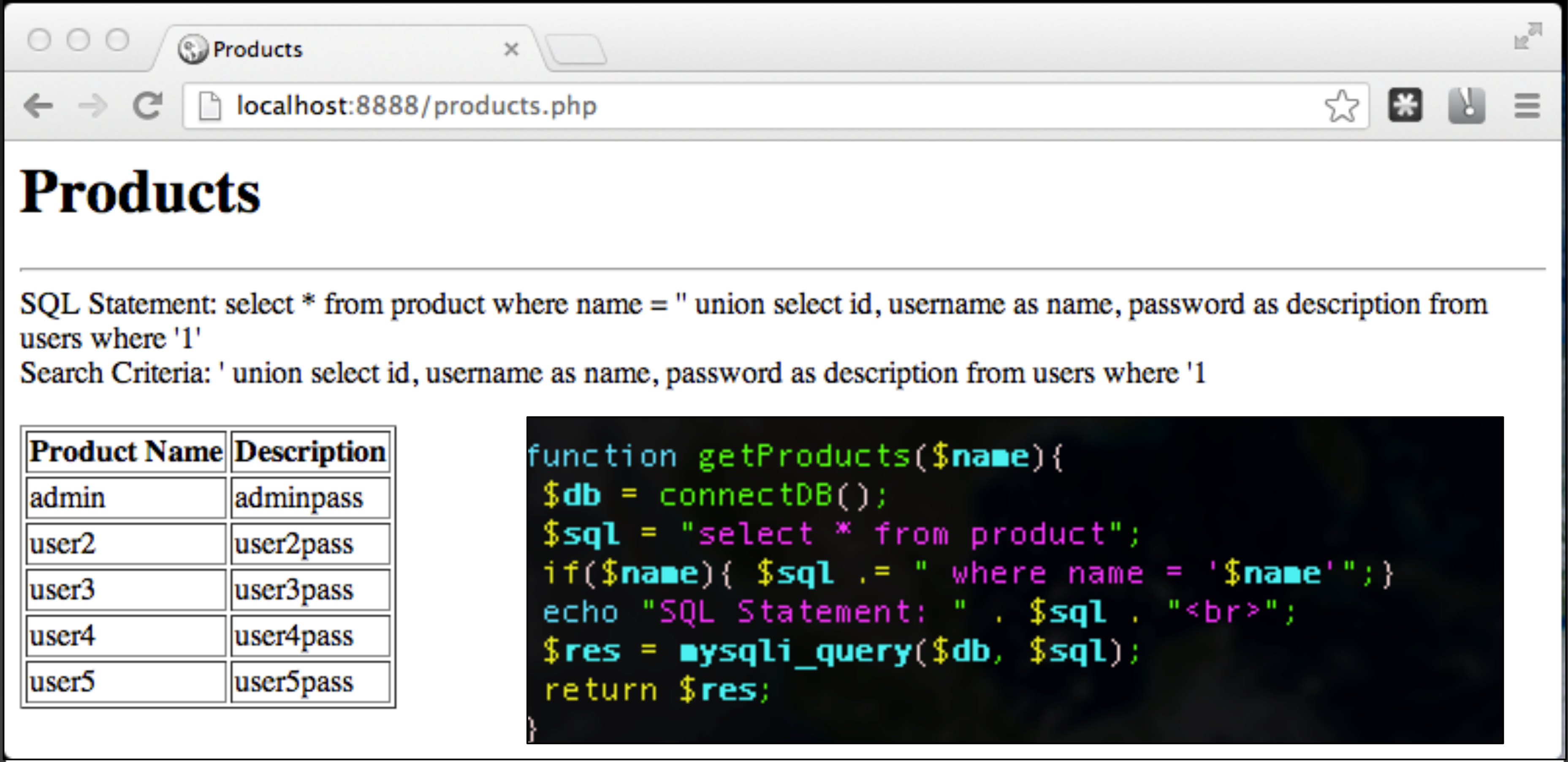


Products

Please indicate which products to display:

Name: ' union select id, username as name, password as description from users where '1

SQL Injection - Dumps Users Table



Products

localhost:8888/products.php

Products

SQL Statement: select * from product where name = '' union select id, username as name, password as description from users where '1'
Search Criteria: ' union select id, username as name, password as description from users where '1

Product Name	Description
admin	adminpass
user2	user2pass
user3	user3pass
user4	user4pass
user5	user5pass

```
function getProducts($name){
 $db = connectDB();
 $sql = "select * from product";
 if($name){ $sql .= " where name = '$name'";}
 echo "SQL Statement: " . $sql . "<br>";
 $res = mysqli_query($db, $sql);
 return $res;
}
```

Mitigation Techniques*

Primary Defenses:

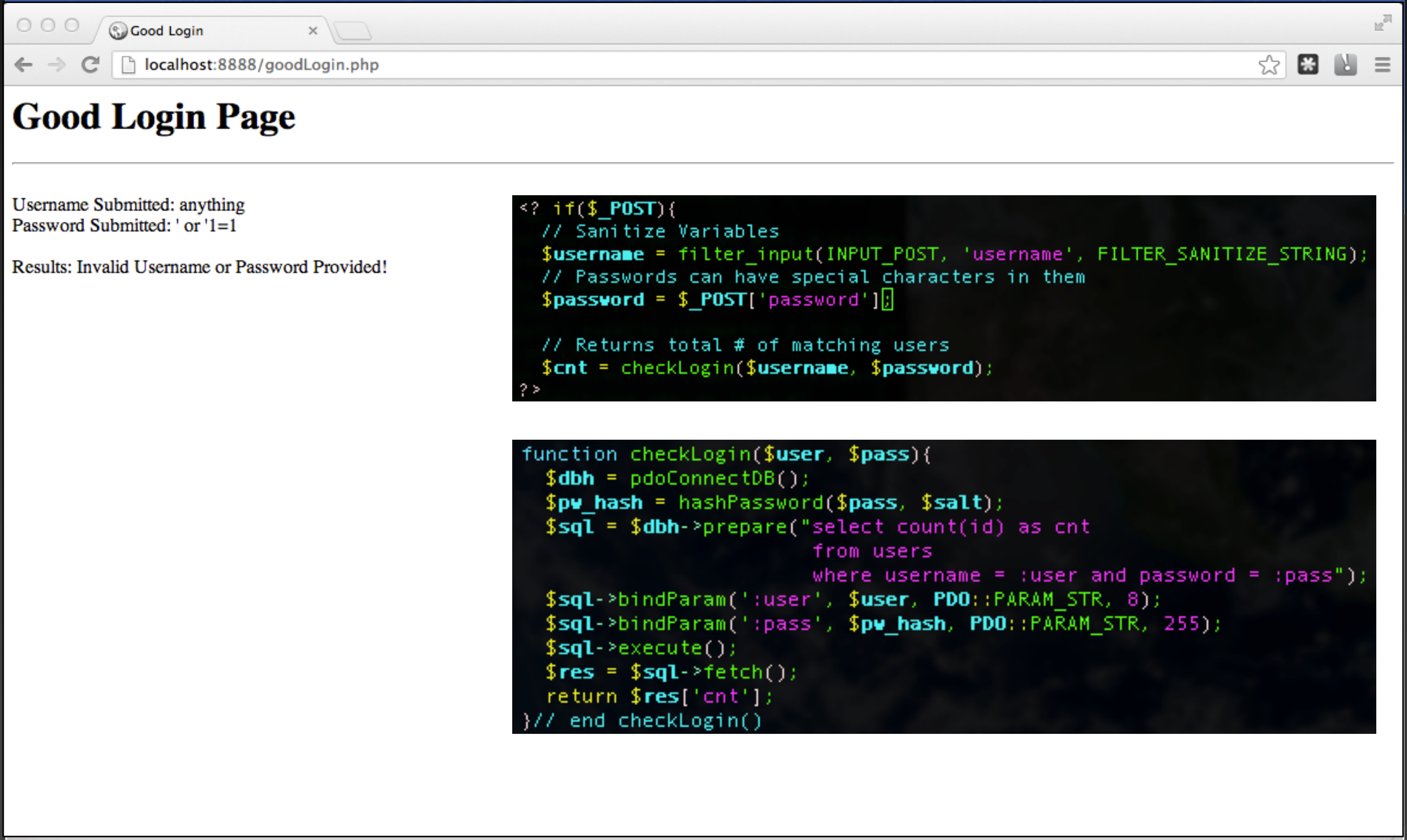
- Use of Prepared Statements (Parameterized Queries)
- Use of Stored Procedures
- Escaping all User Supplied Input

Additional Defenses:

- Least Privilege
- White List Input Validation

Note*: Mitigation techniques recommended by OWASP

Login with Sanitization &Binding



Good Login

localhost:8888/goodLogin.php

Good Login Page

Username Submitted: anything
Password Submitted: ' or '1=1

Results: Invalid Username or Password Provided!

```
<? if($_POST){
  // Sanitize Variables
  $username = filter_input(INPUT_POST, 'username', FILTER_SANITIZE_STRING);
  // Passwords can have special characters in them
  $password = $_POST['password'];

  // Returns total # of matching users
  $cnt = checkLogin($username, $password);
?>
```

```
function checkLogin($user, $pass){
  $dbh = pdoConnectDB();
  $pw_hash = hashPassword($pass, $salt);
  $sql = $dbh->prepare("select count(id) as cnt
                        from users
                        where username = :user and password = :pass");
  $sql->bindParam(':user', $user, PDO::PARAM_STR, 8);
  $sql->bindParam(':pass', $pw_hash, PDO::PARAM_STR, 255);
  $sql->execute();
  $res = $sql->fetch();
  return $res['cnt'];
}// end checkLogin()
```

References

- Comic - https://xkcd.com/327/
- OWASP SQL Injection Prevention Cheat Sheet - https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet
- OWASP Top 10 – https://www.owasp.org
- SQL Injection Definition -http://www.techopedia.com/definition/4126/sql-injection