

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Programų sistemų inžinerijos apsauga

1-asis laboratorinis darbas

Internetinių pažeidžiamų tyrimas ir analizė

Studentas: Lukas Rubikas IFM-5/2 gr.

KAUNAS, 2015

Turinys

Įterpimas.....	3
Aplinka.....	3
Ataka.....	4
Priežastys.....	5
Apsisaugojimas.....	5
XSS.....	6
Aplinka.....	6
Ataka.....	7
Priežastys.....	8
Apsisaugojimas.....	9
Funkcinio lygmens autentifikacijos trūkumas.....	10
Aplinka.....	10
Ataka.....	11
Priežastys.....	14
Apsisaugojimas.....	14
Nuorodos ir šaltiniai.....	15

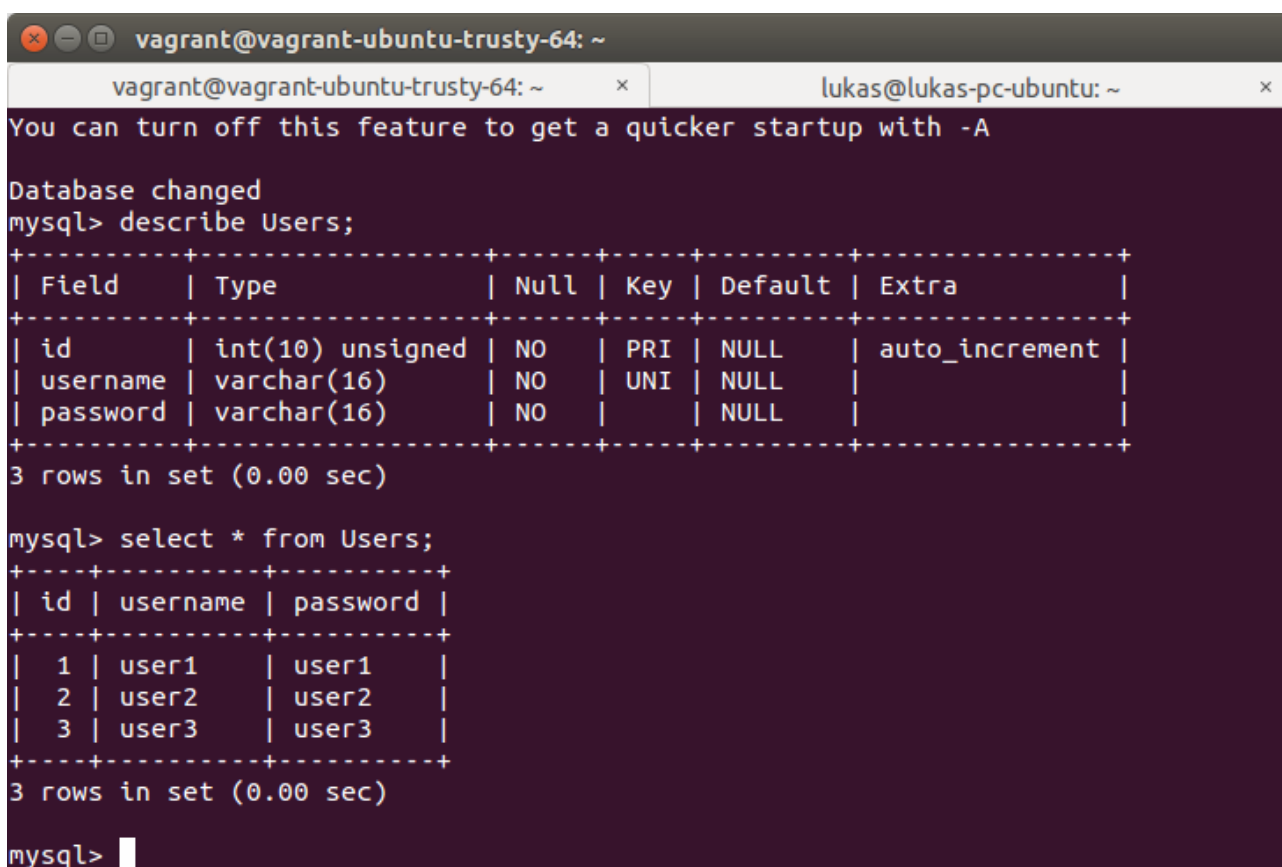
Įterpimas

Įterpimas – labai populiaris ir pavojingas klaidų kategorijos kategorija, kuomet vartotojui leidžiama į SQL, LDAP ir kt. sistemų užklausas įterpti ir savo kodą, taip užklausą modifikuojant, kad ji grąžintų ar gadinėtų įsilaužėliui reikiamą informaciją.

Šiame darbe bus pademonstruota SQL įterpimo ataka.

Aplinka

Ataka bus atlika LAMP (angl. *Linux, Apache, MySQL, PHP*) pakete. MySQL duomenų bazėje egzistuoja viena lentelė *Users*, kurioje saugoma informacija apie sistemos vartotojus. Šiame darbe bus pademonstruota ataka, sugadinanti sistemos SQL užklausos logiką, naudojant patikrinti vartotojo prisijungimo duomenims.



```
vagrant@vagrant-ubuntu-trusty-64: ~
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> describe Users;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id     | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| username | varchar(16)      | NO   | UNI | NULL    |                |
| password | varchar(16)      | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from Users;
+----+-----+-----+
| id | username | password |
+----+-----+-----+
| 1  | user1    | user1    |
| 2  | user2    | user2    |
| 3  | user3    | user3    |
+----+-----+-----+
3 rows in set (0.00 sec)

mysql> 
```

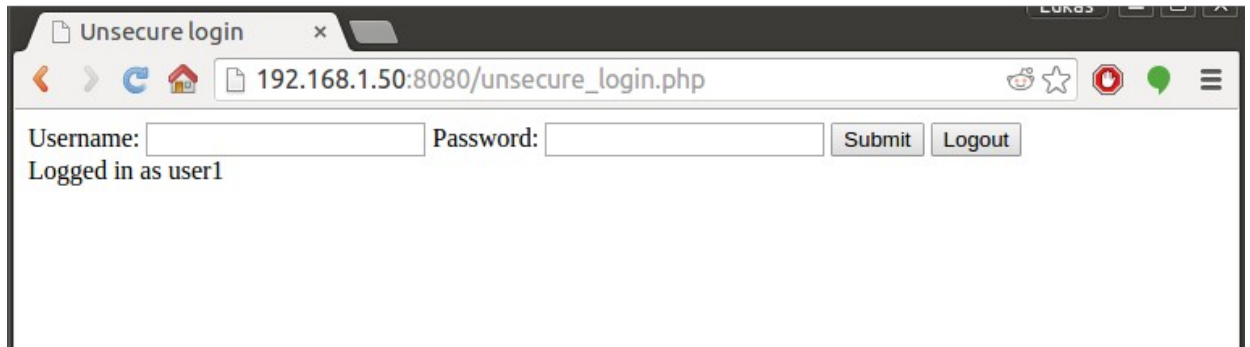
Pastaba: Kadangi tai yra pavyzdinė programa skirta testams, skaidrumo vardan šiame darbe nėra šifruojami slaptažodžiai.

SQL atakai pažeidžiamo kodo ištrauka:

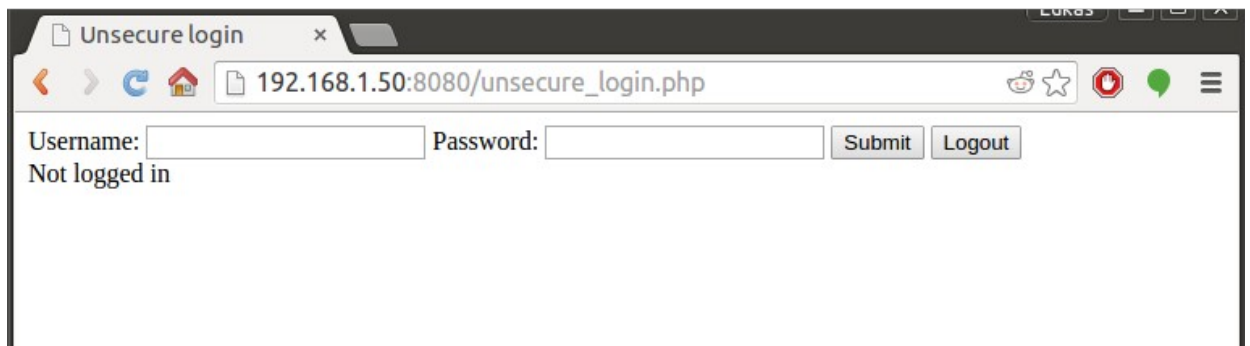
```
$query = "SELECT * FROM Users WHERE username='" . $username . "' AND
        PASSWORD = '" . $password . "' LIMIT 1";
```

Ataka

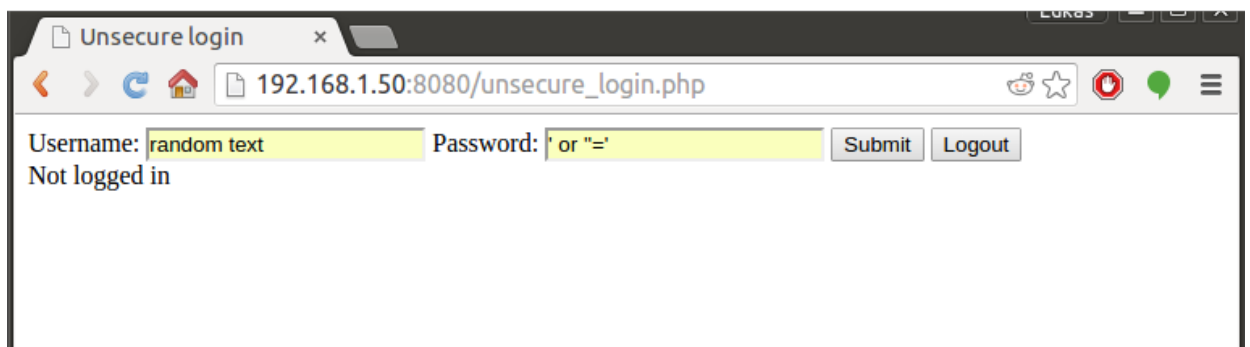
1. Atidarome potencialiai pažeidžiamos įvesties langą:



2. Įterpiame SQL užklausos dalį į vieną iš įvesties laukų:



3. Prisijungimas pavyko.



Priežastys

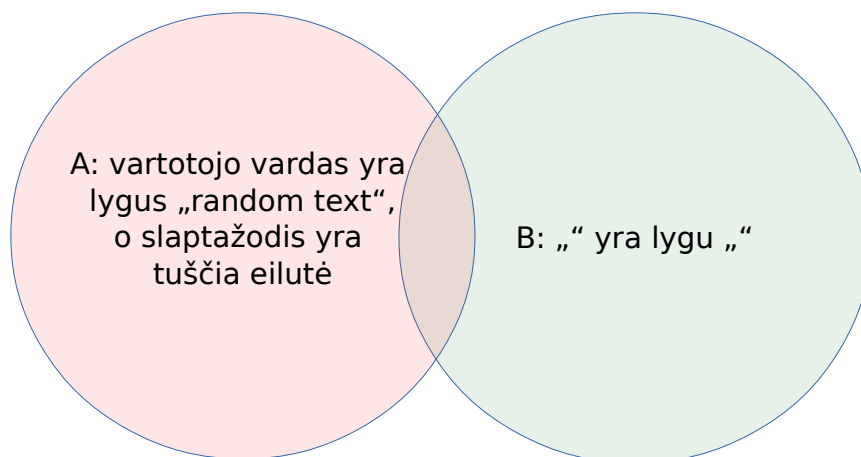
Atlikus SQL injekciją, užbaigta SQL užklausa atrodė šitaip:

```
vagrant@vagrant-ubuntu-trusty-64: ~  
vagrant@vagrant-ubuntu-trusty-64: ~ x vagrant@vagrant-ubuntu-trusty-64: /var/www/... x  
mysql> select * from Users where username = 'random text' and password = '' or '  
' = '' limit 1;  
+-----+-----+-----+  
| id | username | password |  
+-----+-----+-----+  
| 1 | user1    | user1    |  
+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Kaip matome, užklausa grąžino pirmojo lentelės *Users* įrašo duomenis. Tai ypač pavojinga, kadangi pirmieji sistemos vartotojai vartotojų lentelėse dažniausiai būna administratoriai.

Šios atakos priežastys slypi šio SQL sakinio interpretacijoje – kadangi nėra pirmumą įrodančių skliaustų, sąlyga OR turi pirmenybę prieš sąlygą AND ir kadangi dešinioji sąlygos OR pusė yra tiesa (tuščia eilutė ("")) yra lygi kitai tuščiai eilutei), galima vykdyti SELECT užklausos sakinio dalį. Kitaip tariant, užklausa grąžins rezultatą, jei:

$$S = A \vee B$$



Apsisaugojimas

Geriausia SQL atakų apsisaugojimo priemonė – parametrizacija arba paruoštosios užklausos. Paruošiamųjų užklausų metu, duomenų bazėje pranešama apie “artėjančią” jau sukonkretizuotą užklausą, į kurios laukelių vietas ateis vartotojų apibrėžti parametrai. Jeigu programiniame kode naudojama parametrizacija, SQL užklausų bandymai bus interpretuoti tik kaip užklausų parametrai, o ne kaip užklausos dalis.

```
txtUserId = getRequestString("UserId");  
txtSQL = "SELECT * FROM Users WHERE UserId = @0";  
db.Execute(txtSQL, txtUserId);
```

XSS

Šios klaidos atsiranda, kai programa leidžia išsaugoti ir parodyti naršyklėje neišvalytus duomenis (dažnai su JavaScript kodu). Tokiu būdu galima mėginti savintis vartotojo sausainėlius (cookies), vykdyti peradresavimą.

Aplinka

Ataka bus taip pat atlika LAMP (angl. *Linux, Apache, MySQL, PHP*) pakete. MySQL duomenų bazėje pridedama dar viena lentelė *Comments*, kurioje saugoma informacija apie sistemos vartotojų komentarus. Šiame darbe bus pademonstruota ataka, įterpianti JavaScript kodą į komentaro laukelį, o kai komentaras atvaizduojamas, kliento lygmenyje JavaScript kodas įsivykdo pagal atakuotojo kėslus.

```
lukas@lukas-pc-ubuntu: ~/Documents/Studies/T120M14... x | vagrant@vagrant-ubuntu-trusty-64: ~ x
mysql> describe Comments;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)   | NO   | PRI | NULL    | auto_increment |
| author | varchar(16) | YES  |     | NULL    |              |
| comment | text      | YES  |     | NULL    |              |
| datetime | datetime  | YES  |     | NULL    |              |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from Comments;
+-----+-----+-----+-----+
| id | author | comment | datetime |
+-----+-----+-----+-----+
| 1 | user1 |  | 2015-11-28 17:21:06 |
| 2 | user1 | Komentaras | 2015-11-28 17:24:22 |
| 3 | user1 | sample | 2015-11-28 17:35:11 |
| 4 | user1 | sample | 2015-11-28 17:35:33 |
| 5 | user1 | sample | 2015-11-28 17:35:49 |
| 6 | user1 | sample | 2015-11-28 17:36:15 |
| 7 | user2 | another sample | 2015-11-28 17:39:16 |
| 8 | user2 | window.alert("annoying pop-up"); | 2015-11-28 17:39:34 |
| 9 | user2 | <script>window.alert("Annoying pop-up");</script> | 2015-11-28 17:40:26 |
+-----+-----+-----+-----+
9 rows in set (0.00 sec)

mysql>
```

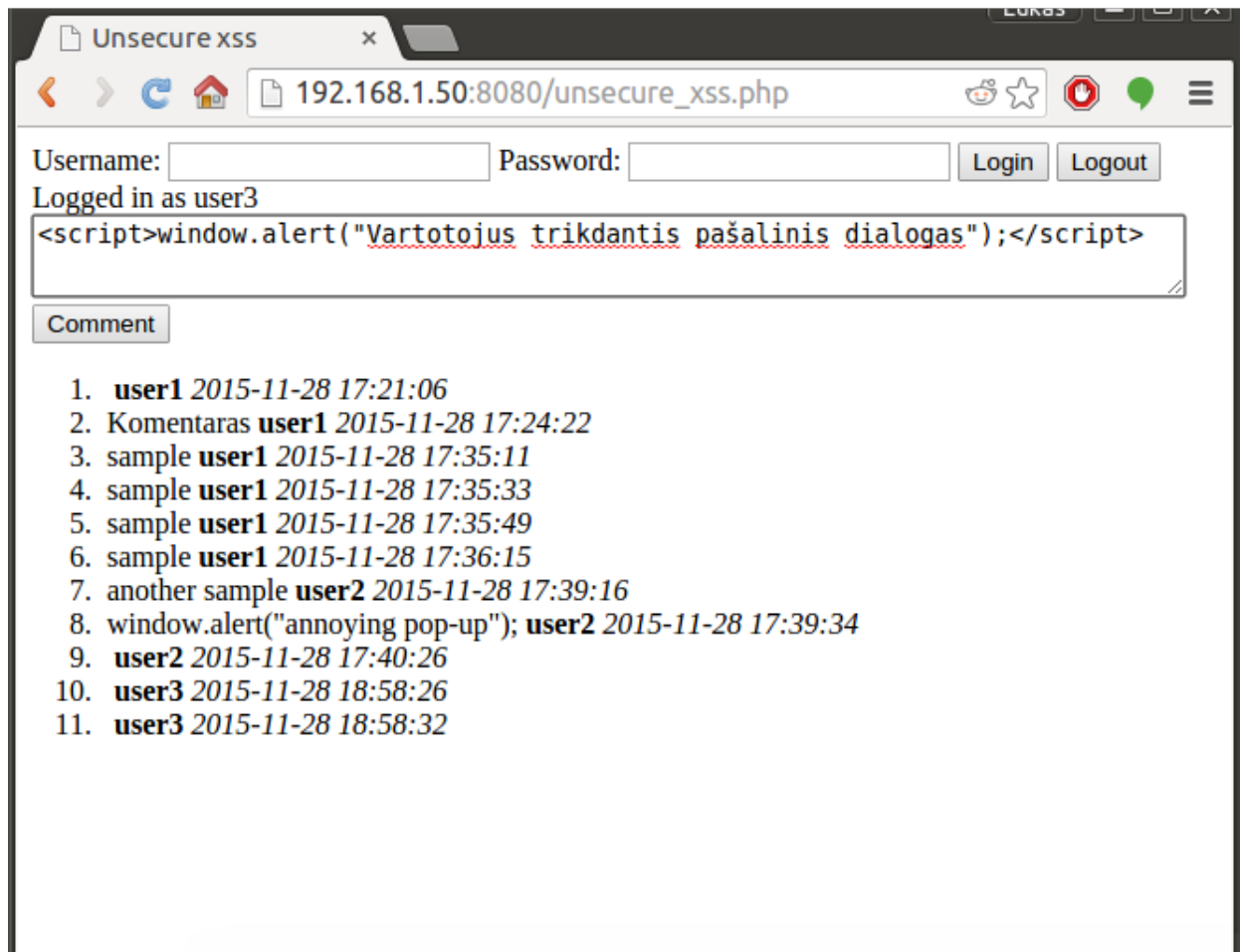
XSS atakai pažeidžiamo kodo ištrauka:

```
<div class="Comments">
  <ol>
    <?php
      while ($commentsRow = mysql_fetch_assoc($commentsResult)) {
        echo "<li><span>" . $commentsRow["comment"] . "</span>
<strong>" .
          $commentsRow["author"] . "</strong> <i>" .
          $commentsRow["datetime"]
            . "</i></li>";
      }
    ?>
```

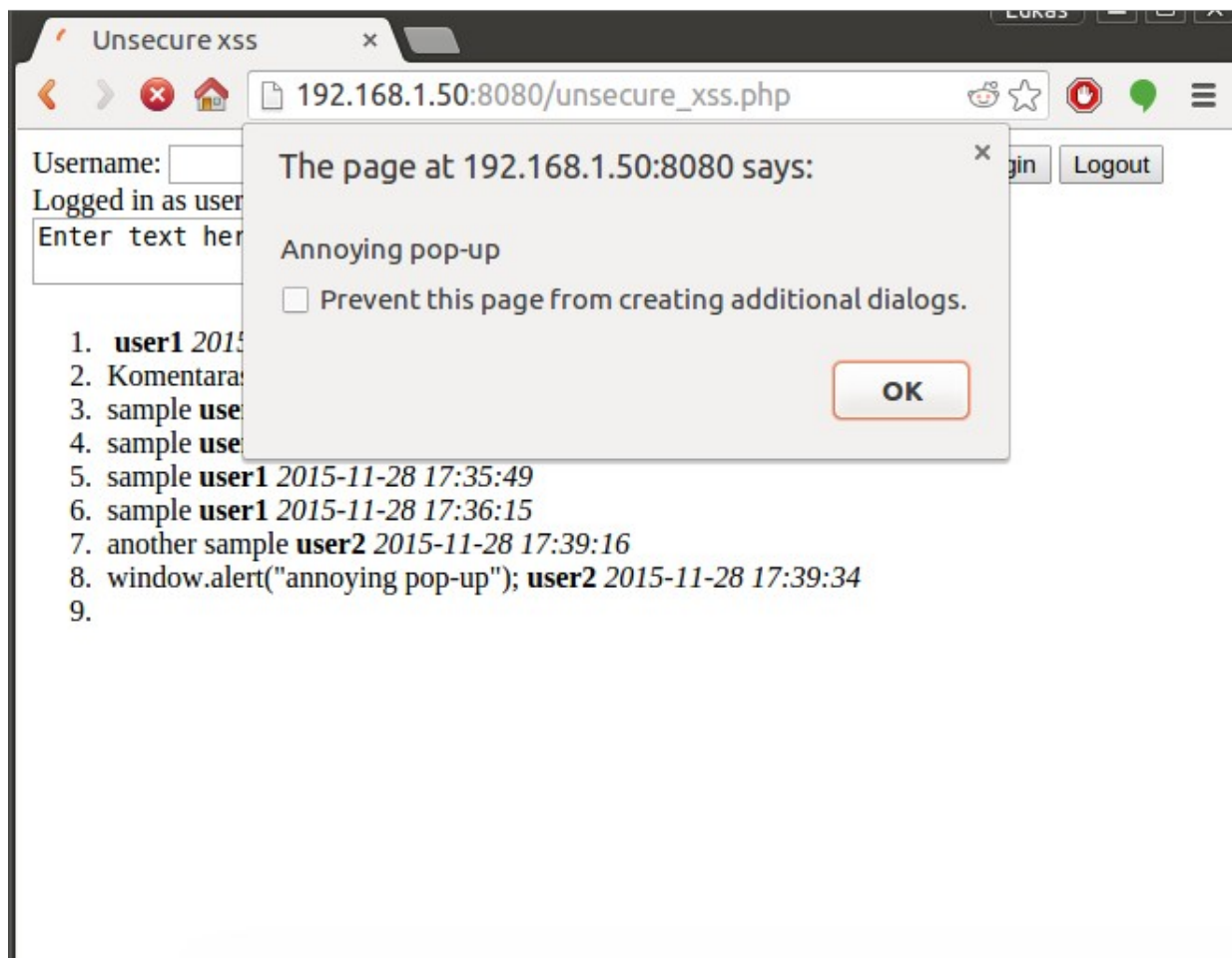
```
</ol>  
</div>
```

Ataka

1. Prisijungę kaip vartotojas, įvedame JavaScript kodą į komentaro laukelį:



2. Patvirtinę komentarą, galime išvysti praėjusio bandymo (kuomet į window.alert() funkcijos parametą buvo įrašyta žinutė "Annoying pop-up") vaisius:



3. Duomenų bazėje lentelė Comments atrodo šitaip:

```

mysql> select * from Comments;
+----+-----+-----+-----+
| id | author | comment | datetime |
+----+-----+-----+-----+
| 1 | user1 | Komentaras | 2015-11-28 17:21:06 |
| 2 | user1 | sample | 2015-11-28 17:24:22 |
| 3 | user1 | sample | 2015-11-28 17:35:11 |
| 4 | user1 | sample | 2015-11-28 17:35:33 |
| 5 | user1 | sample | 2015-11-28 17:35:49 |
| 6 | user1 | sample | 2015-11-28 17:36:15 |
| 7 | user2 | another sample | 2015-11-28 17:39:16 |
| 8 | user2 | window.alert("annoying pop-up"); | 2015-11-28 17:39:34 |
| 9 | user2 | <script>window.alert("Annoying pop-up");</script> | 2015-11-28 17:40:26 |
| 10 | user3 | <script>windows.alert("Vartotojus trikdantis paÅ;alini dialogas");</script> | 2015-11-28 18:58:26 |
| 11 | user3 | <script>windows.alert("Vartotojus trikdantis paÅ;alini dialogas");</script> | 2015-11-28 18:58:32 |
| 12 | user3 | <script>window.alert("Vartotojus trikdantis paÅ;alini dialogas");</script> | 2015-11-28 19:00:34 |
| 13 | user3 | <script>window.alert("Vartotojus trikdantis paÅ;alini dialogas");</script> | 2015-11-28 19:00:53 |
+----+-----+-----+-----+
13 rows in set (0.00 sec)

mysql>

```

Priežastys

Iš duomenų bazės gaunami duomenys atvaizduojami HTML žymėjimo kalbos kontekste. Jeigu duomenys nėra kaip nors užkoduojami arba specialūs HTML simboliai pakeičiami ("brackets", arba "<>"), komentaruose taip pat galima rašyti žiniatinklio kodą HTML kalba. Kadangi <script> yra HTML elementas, tarp kurių skliaustų (angl.

Brackets) interpretuojama JavaScript kalba, komentatoriaus įrašytas JavaScript kodas yra interpretuojamas.

Apsisaugojimas

Apsisaugoti nuo XSS atakos galima pakeičiant komentarų specialius HTML simbolius į jų kodų atitikmenis. Vartotojui jie bus atvaizduojami taisyklingai, bet naršyklė jų nebeinterpretuos kaip HTML kodo dalies.

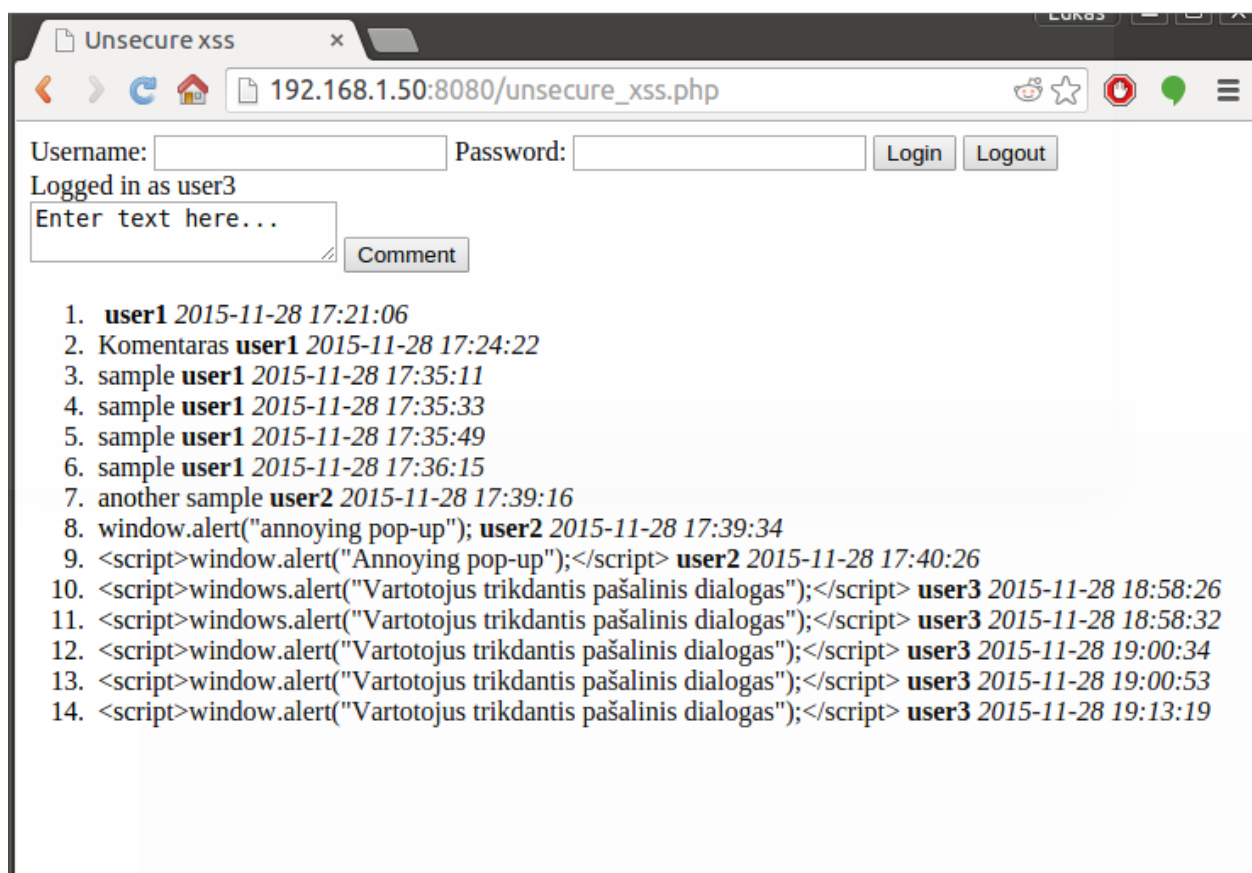
Kodo eilutė:

```
echo "<li><span>" . $commentsRow["comment"] . "</span> <strong>" .
```

Pakeičiama į:

```
echo "<li><span>" . htmlspecialchars($commentsRow["comment"]) .  
"</span> <strong>" .
```

Duomenų bazėje saugojami komentarai nepakeičiami, bet atvaizduojami be atakos požymių:



Funkcinio lygmens autentifikacijos trūkumas

Kiekviena "iš išorės" pasiekiamą PL funkciją turi turėti prieinamumo kontrolę. Pvz. tikrinama ar vartotojas gali pasiekti "admin" langą, tačiau jame vykdomos AJAX tipo užklausos, kuriuose jau nebetikrinama ar tai reikiamo lygio vartotojas.

Aplinka

Ši ataka, kaip ir praėjusios dvi, bus taip pat atlika LAMP (angl. *Linux, Apache, MySQL, PHP*) pakete. Šiame darbe bus pademonstruota ataka, kuomet komentarų ištrynimo galimybė matoma tik juos sukūrusiems vartotojams, tačiau vartotojui pakeitus užklausos informaciją, dėl neegzistuojančios funkcinio lygmens autentifikacijos ir autorizacijos, ištrinamas kito autoriaus komentaras.

Atakai pažeidžiamo kodo fragmentas:

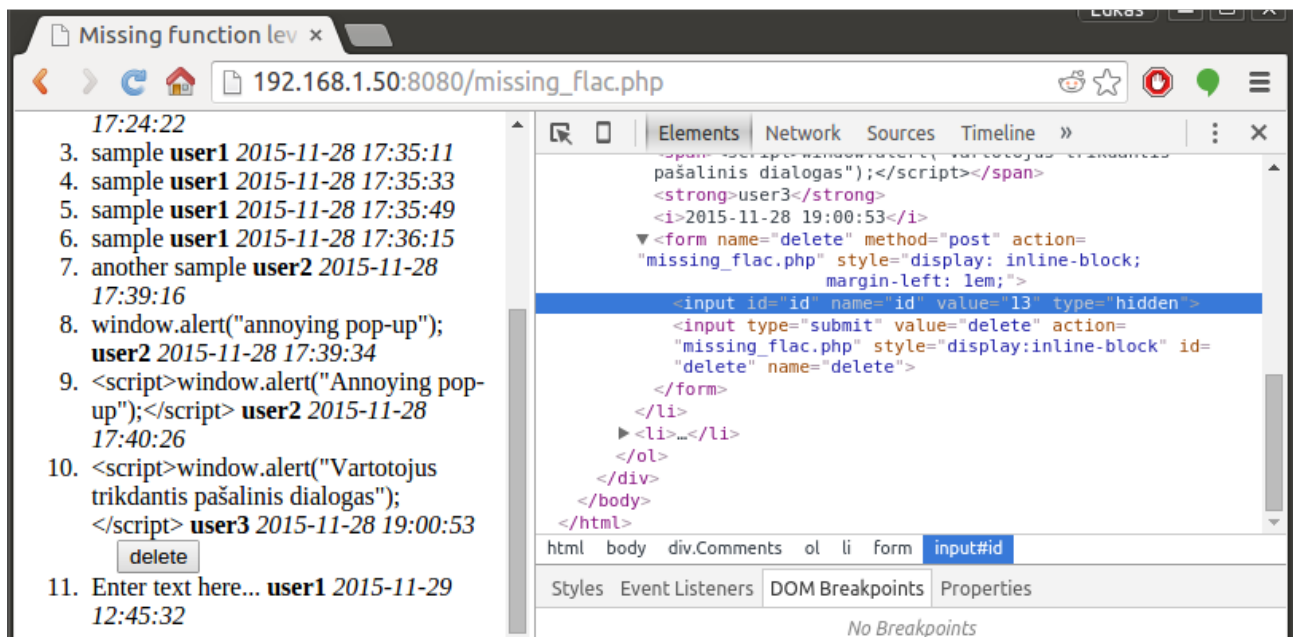
```
9     if(isset($_POST['delete'])) {
10         $conn = mysql_connect($dbhost, $dbuser, $dbpass);
11         $id = $_POST['id'];
12         if(!$conn) {
13             $errorMessage = "Could not connect";
14             die('Could not connect: ' . mysql_error());
15         }
16         $db_selected = mysql_select_db($dbname, $conn);
17         if (!$db_selected) {
18             $errorMessage = "Can't use database";
19             die ('Can\'t use database : ' . mysql_error());
20         }
21         $query = "DELETE FROM Comments WHERE id = '" . $id . "'";
22         $result = mysql_query($query);
23         if (!$result) {
24             $errorMessage = "Invalid query";
25             die ('Invalid query : ' . mysql_error());
26         }
27     }
```

Ataka

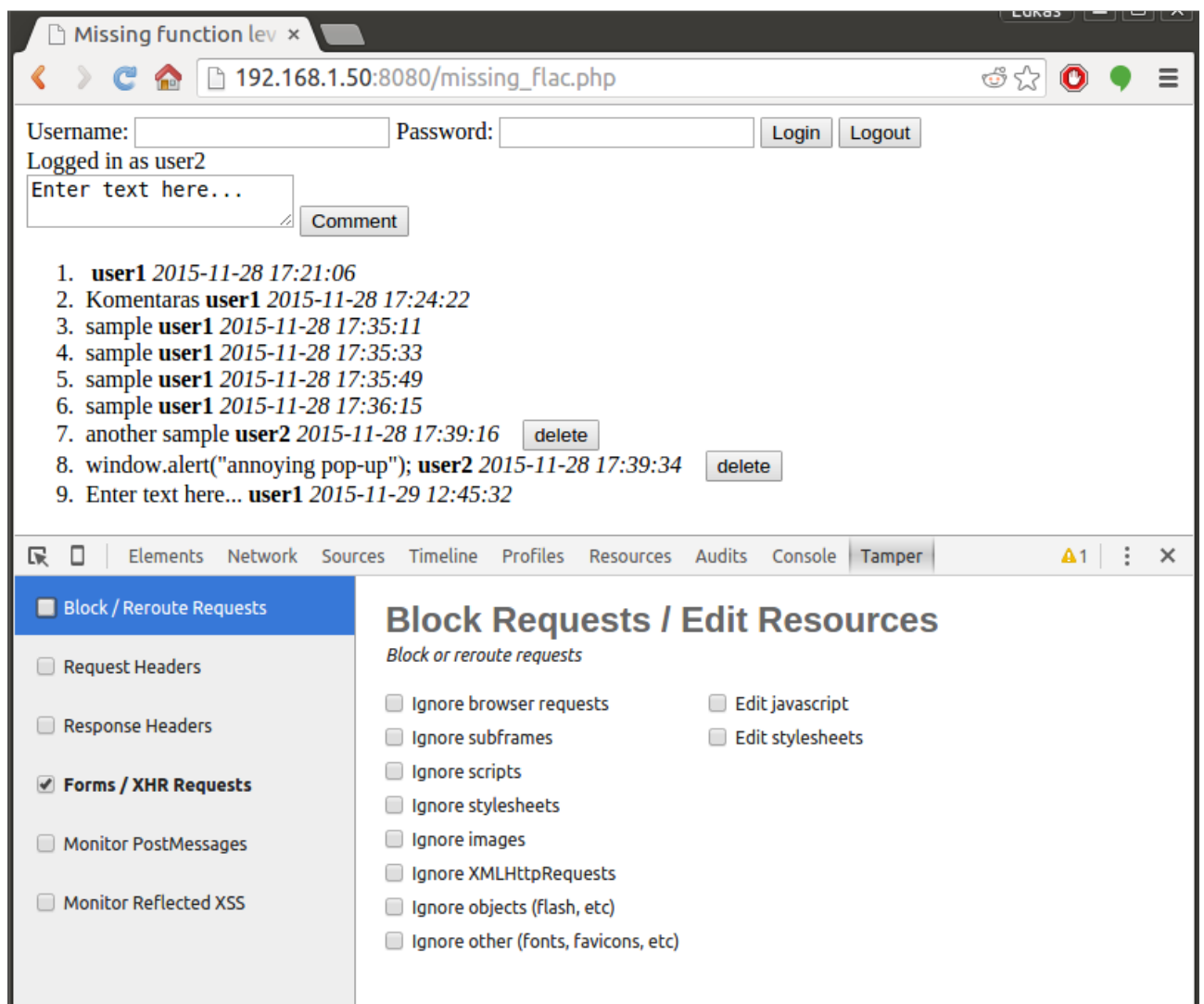
1. Prisijungiame kaip sistemos vartotojas. Matome galimybę ištrinti savo paties parašytą komentarą:



2. Pasinaudojame naršyklės *inspect* įrankiu ir peržiūrime mygtuko elementą:



3. Matome, kad kiekvienas mygtukas šalia turi paslėptą įvesties laukelį, veikiausiai nurodantį komentaro identifikuojantį numerį.
4. Įsirašome bet kokią užklausą redaguoti galintį naršyklės plėtinį. Šiame darbe naudosime "Tamper Chrome".
5. Pažymime varnelę "Forms/XHR":



6. Išsaukiame užklausą, paspaudę mygtuką *delete* prie savo įrašyto komentaro.
7. Atsidariusiame lange pakeičiame *id* į, pavyzdžiui, 1, ir paspaudžiame *Send*:
8. Kaip matome, buvo ištrintas pirmasis komentaras, parašytas 2015-11-28 17:21:06, vartotojo *user1*!

Tamper Chrome (application)

Type	Method	URI	Extra
form	post	http://192.168.1.50:8080/missing_flac.php	

Request Details

Request

Method

post

Action

http://192.168.1.50:8080/missing_

Form fields

id

7

delete

delete

[new]

Decoder ▲

Send

Block

Filter request list:

Clear requests

☐ debug injection
 ☐ ignore requests

Tamper Chrome (application)

Type	Method	URI	Extra
form	post	http://192.168.1.50:8080/missing_flac.php	

Request Details

Request

Method

post

Action

http://192.168.1.50:8080/missing_

Form fields

id

1

delete

delete

[new]

Decoder ▲

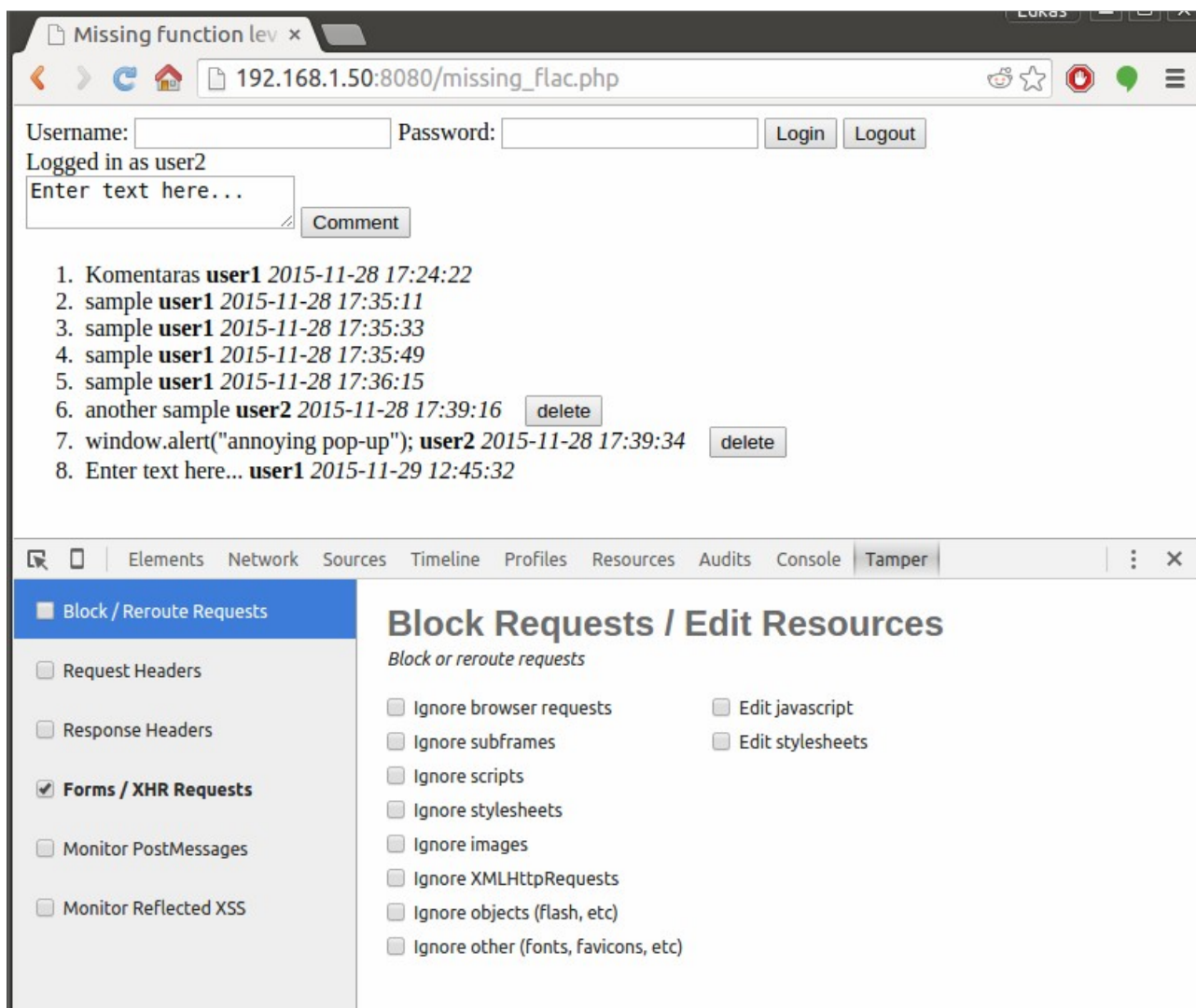
Send

Block

Filter request list:

Clear requests

☐ debug injection
 ☐ ignore requests



Priežastys

Buvo padaryta klaidinga programuotojo prielaida, kad jeigu jis vienintėlis žinos savo komentaro identifikacijos numerį, jis vienintėlis galės jį ištrinti, tačiau nepagalvojo, kad atakuotojas svetimo komentaro identifikacijos numerį gali atspėti ir manipuliudamas užklausomis jį ištrinti.

Apsisaugojimas

Nuo šios atakos būtų galima apsisaugoti komentaro ištrynimo užklausoje patikrinant, ar prisijungęs vartotojas taip pat yra ir komentaro autorius:

```
21 $query = "DELETE FROM Comments WHERE id = '" . $id . "' AND author = '" .
22 $_SESSION['user']['username'] . "'";
```

Nuorodos ir šaltiniai

1. XSS Skype iOS ir Windows platformose:
 1. <https://superevr.com/blog/2011/xss-in-skype-for-ios>
 2. <http://techcrunch.com/2011/09/20/skype-aware-of-xss-vulnerability-in-ios-apps-working-hard-to-fix-it/>
 3. <http://blogs.skype.com/2011/07/15/explaining-the-cross-site-scri/>
2. XSS Youtube vaizdo peržiūrų svetainėje:
 1. <http://www.acunetix.com/blog/articles/dangerous-xss-vulnerability-found-on-youtube-the-vulnerability-explained/>