TK2100: Informasjonssikkerhet
Lesson 10: XSS

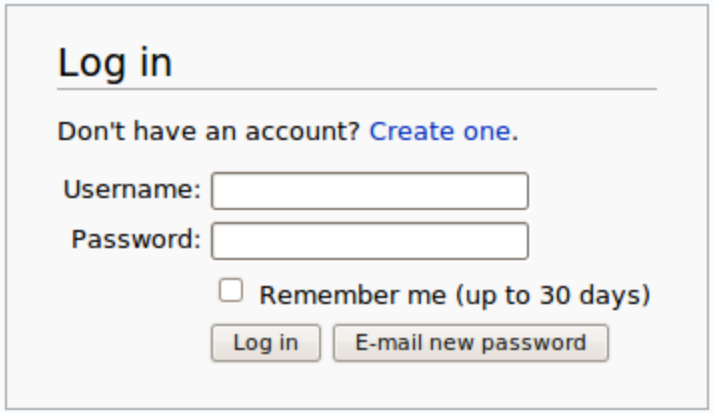
Dr. Andrea Arcuri
Westerdals Oslo ACT
University of Luxembourg

Goals

- Understand why some special characters need to be escaped/sanitized
- Understand how XSS attacks can be carried out
- Learn that you should *never trust user inputs*

Data Escaping/Sanitization

HTML Form Data



- How is data sent in a HTML Form?
- What is the structure of payload of the HTTP POST request?
- JSON? eg *{“username”:“foo”, “password”:123}*
- XML? eg *<data><username>foo</username><password>123</password></data>*

x-www-form-urlencoded

- For textual data, like inputs in a HTML form
  - For binary data like file uploads, can use *multipart/form-data*
- Old format which is part of the HTML specs
  - *https://www.w3.org/TR/html/sec-forms.html#urlencoded-form-data*
- Each form element is represented with a pair *<name>=<value>*, where each pair is separated by a **&**
- Eg.: *username=foo**&**password=123*

What if values contain “=” or “&”?

- Eg, password: “123&bar=7”
- (Wrong) result: username=foo&**password=123**&*bar=7*
- The “*bar=7*” would be wrongly treated as a third input variable called “*bar*” with value “*7*”, and not be part of the “password” value

Solution: Special Encoding

- Stay same: “*”, “-”, “.”, “_”, 0-9, a-z, A-Z
- Space “ ” becomes a “+”
- The rest become “%HH”, a percent sign and two hexadecimal digits representing the code of the character (default UTF-8)
- So, “123&bar=7” becomes “123**%26**bar**%3D**7”
- %26 = (2*16)+6 = 38, which is the code for **&** in ASCII
- %3D = (3*16)+13 = 61, which is the code for **=** in ASCII
  - Recall, hexadecimal D=13 (A=10,…, F=15)

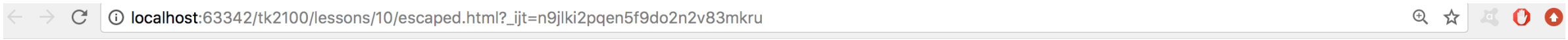
But…

- What if I have a “%” in my values? Would not that mess up the decoding?
- E.g, password=“%3D”, don’t want to be wrongly treated as a “=”
- Not an issue, as symbol “%” is encoded based on its ASCII code 37, ie “*%25*3D”
  - %25 = (2*16)+5 = 37

Text Transformations

- We can represent text in various formats, eg, HTML, XML, JSON, x-www-form-urlencoded
- Such formats use special symbols to define *structures* of the document
  - eg = and & in HTML form data, and <> in HTML/XML documents
- Input text values should NOT use those special structure/syntax symbols
- Need to be *transformed* (aka *escaped*) into non structure symbols
  - & into %26, and = into %3D in HTML form data

What About HTML???



How to represent the symbols of a tag with attribute without getting them interpreted as HTML tags?
For example:
Foo
vs.
Foo

HTML/XML Escaping

- “**&**” followed by name (or code), closed by “**;**”
- **"** for **“** (double quotation mark)
- **&** for **&** (ampersand)
- **'** for **‘** (apostrophe)
- **<** for **<** (less-than)
- **>** for **>** (greater-than)
- These are most common ones

See “escaped.html” file

Foo

vs.

Foo

What actually needs to be escaped depends on context

- <**div id=""<p>"**">
  "**<**p**>**"
  </**div**>
- Representing **“<p>”** (quotes included)
- In attributes, quotes “ need to be escaped (**"**), but no need there for **<>**, as those latter are no string delimiters
- In node content, it is the other way round

XSS

User Content

- Text written by user which is displayed in the HTML pages when submitted (eg HTML form)
  - eg, Chats and Discussion Forums
  - but also showing back the search query when doing a search
- Also query parameters in URLs are a form of user input if crafted by an attacker
  - eg, *www.foo.com?x=10* if then value of *x* is displayed in the HTML
  - recall, attacker can use social engineering to trick user to click on a link
- *What is the most important rule regarding user content given as input to a system???*

NEVER TRUST USER INPUTS!!!

NEVER

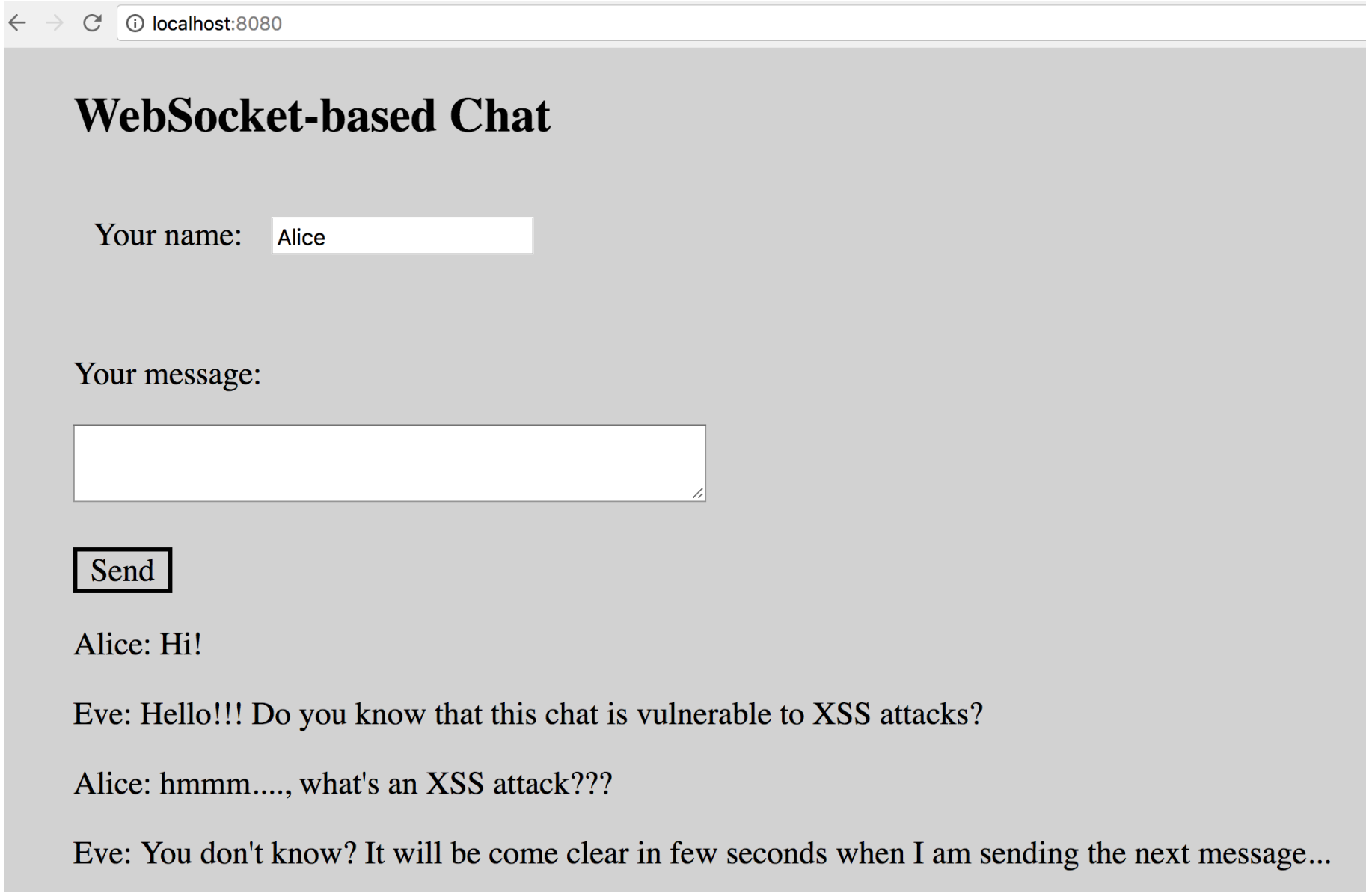
TRUST

USER

INPUTS!!!

NEVER TRUST USER INPUTS!!!

But Why???


localhost:8080
WebSocket-based Chat
Your name:
Alice
Your message:
Send
Alice: Hi!
Eve: Hello!!! Do you know that this chat is vulnerable to XSS attacks?
Alice: hmmm...., what's an XSS attack???
Eve: You don't know? It will be come clear in few seconds when I am sending the next message...

After Eve’s message, chat program is gone on Alice’s browser…

What was the problem?

```
function updateMessages(dto){

    var msgDiv = document.getElementById('messagesId');

    var p = "<p>" + dto.author + ": " + dto.text + "</p>";

    msgDiv.innerHTML += p;
}
```

String Concatenation

- **var** p = " " + dto.**author** + ": " + dto.**text** + " ";
- Should **NEVER** concatenate strings directly to generate HTML when such data comes from user
  - ie, that is a very, very bad example of handling user inputs
- If data is not escaped, could have HTML <tags> that are interpreted by browser as HTML commands
- Could execute JavaScript!!! And so do whatever you want on a page
- Eg., *dto.text = "<script>…</script>"*

Cross-site Scripting (XSS)

- Type of attack in which malicious JavaScript is injected into a web page
- One of the most common type of security vulnerability on the web
- Typically exploiting lack of escaping/sanitization of user inputs when generating HTML dynamically (both client and server side)

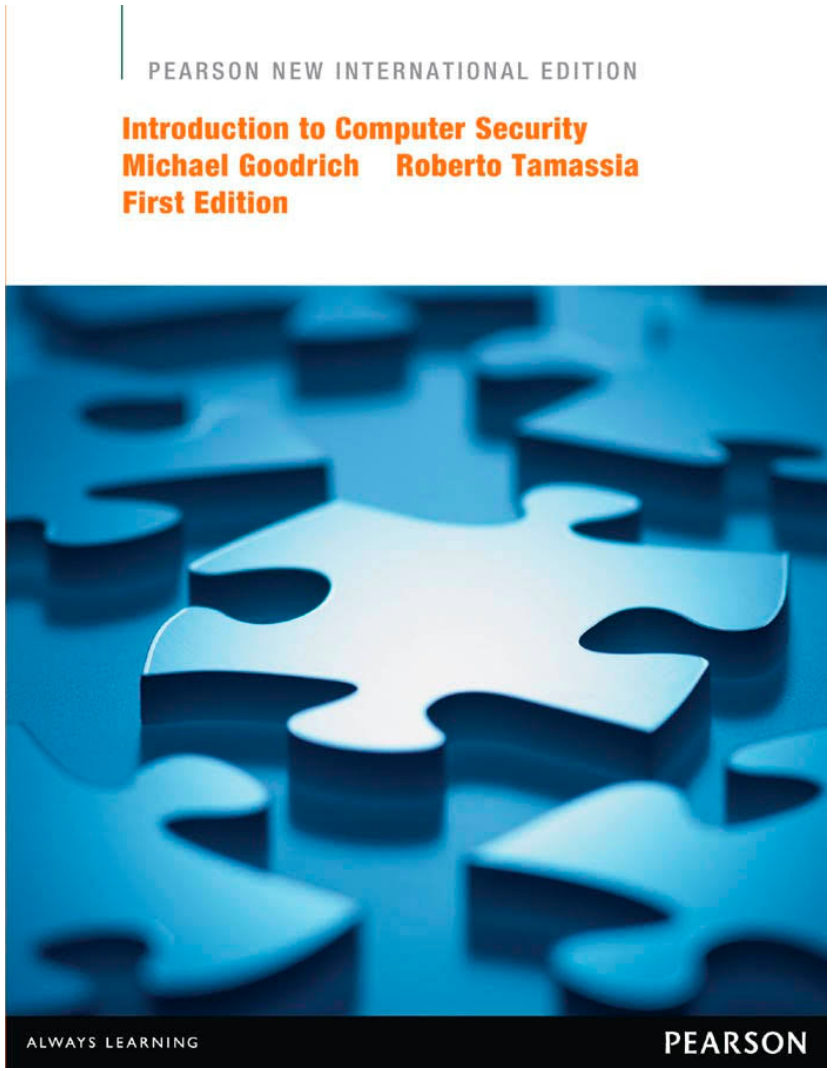
Browser Security

- Most browsers will not execute any *<script>* block that has been dynamically added to the page
  - eg, when changing the HTML by altering *"innerHTML"*
- But that is simply futile… because you can still create HTML tags with JS handlers that are executed immediately
- *<img src='aURLthatNotExist' **onerror**="… JS here…">*

What To Do?

- When dealing with user inputs, always need to escape/sanitize them before use
- This applies both client-side (JS) and server-side (Java, PHP, C#, etc.)
- There are many edge cases, so must use an *existing* library to sanitize the inputs
  - This will depend on the programming language and framework
  - Do NOT write your own escape/sanitize functions

For Next Week



- Book pages: 357-363
- Note: when I tell you to **study** some specific pages in the book, it would be good if you also *read* the other pages in the same chapter at least once
- Exercises for Lesson 10 on GitHub repository