SENG 360 - Security Engineering
Web Security - XSS

Jens Weber

Fall 2021

University of Victoria

Recall from last classes

2021 List

Rank	ID	Name	Score	2020 Rank Change
[1]	CWE-787	Out-of-bounds Write	65.93	+1
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	46.84	-1
[3]	CWE-125	Out-of-bounds Read	24.9	+1
[4]	CWE-20	Improper Input Validation	20.47	-1
[5]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	19.55	+5
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	19.54	0
[7]	CWE-416	Use After Free	16.83	+1
[8]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.69	+4
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	14.46	0
[10]	CWE-434	Unrestricted Upload of File with Dangerous Type	8.45	+5
[11]	CWE-306	Missing Authentication for Critical Function	7.93	+13
[12]	CWE-190	Integer Overflow or Wraparound	7.12	-1
[13]	CWE-502	Deserialization of Untrusted Data	6.71	+8

Learning Objectives

At the end of this class you will be able to

- Describe cross-site scripting vulnerabilities and mitigations
- Explain difference between reflected, stored, blind and DOM-based XSS

University of Victoria

© Jens H. Weber

What is Cross-Site Scripting (XSS)?

XSS attacks enable attackers to inject client-side scripts into web pages viewed by other users. [Wikipedia]

Four types:

1. Reflected XSS
2. Persistent XSS
3. Blind XSS
4. DOM-based XSS

University of Victoria

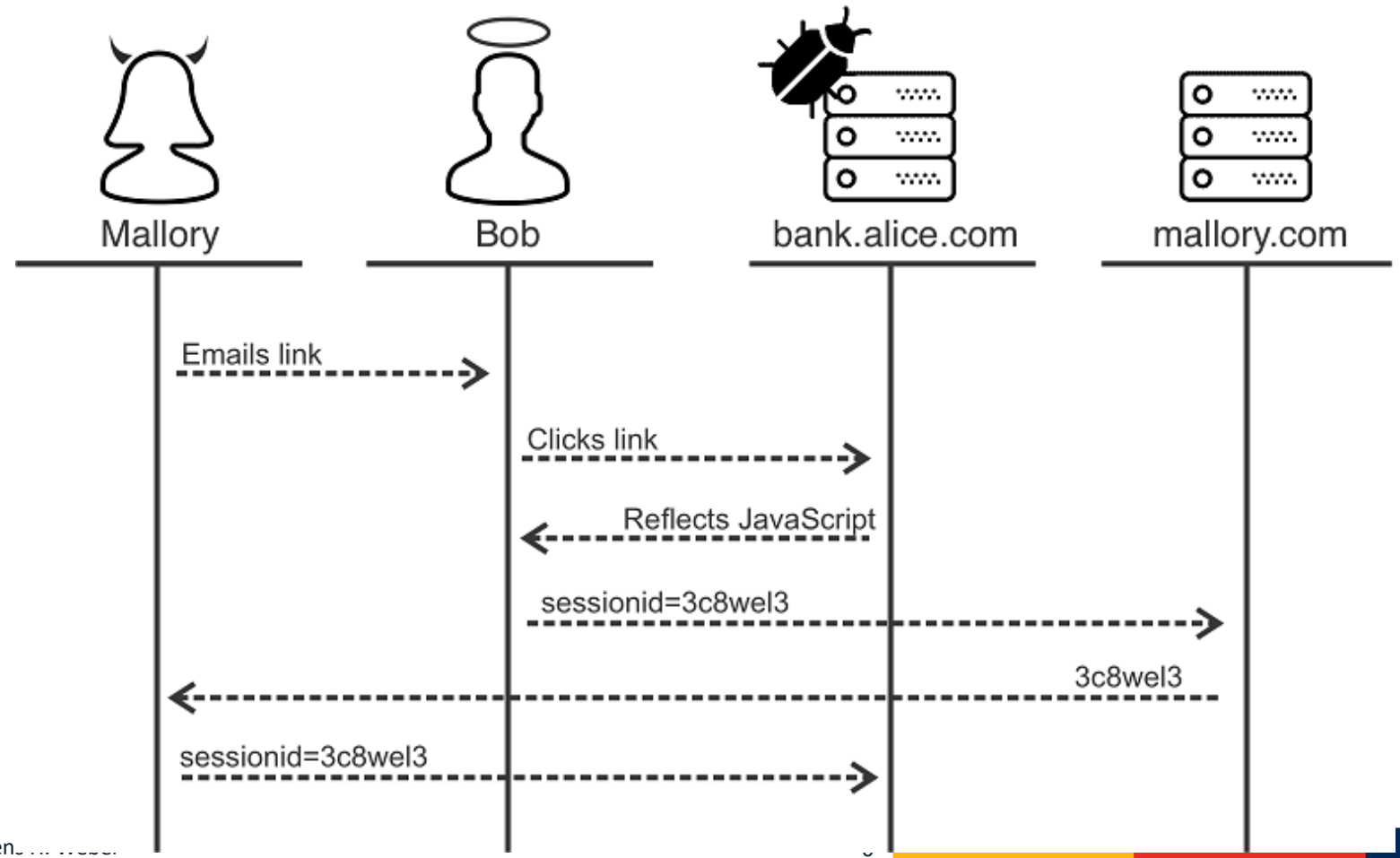
© Jens H. Weber

Reflected XSS

University of Victoria

© Jens H. Weber

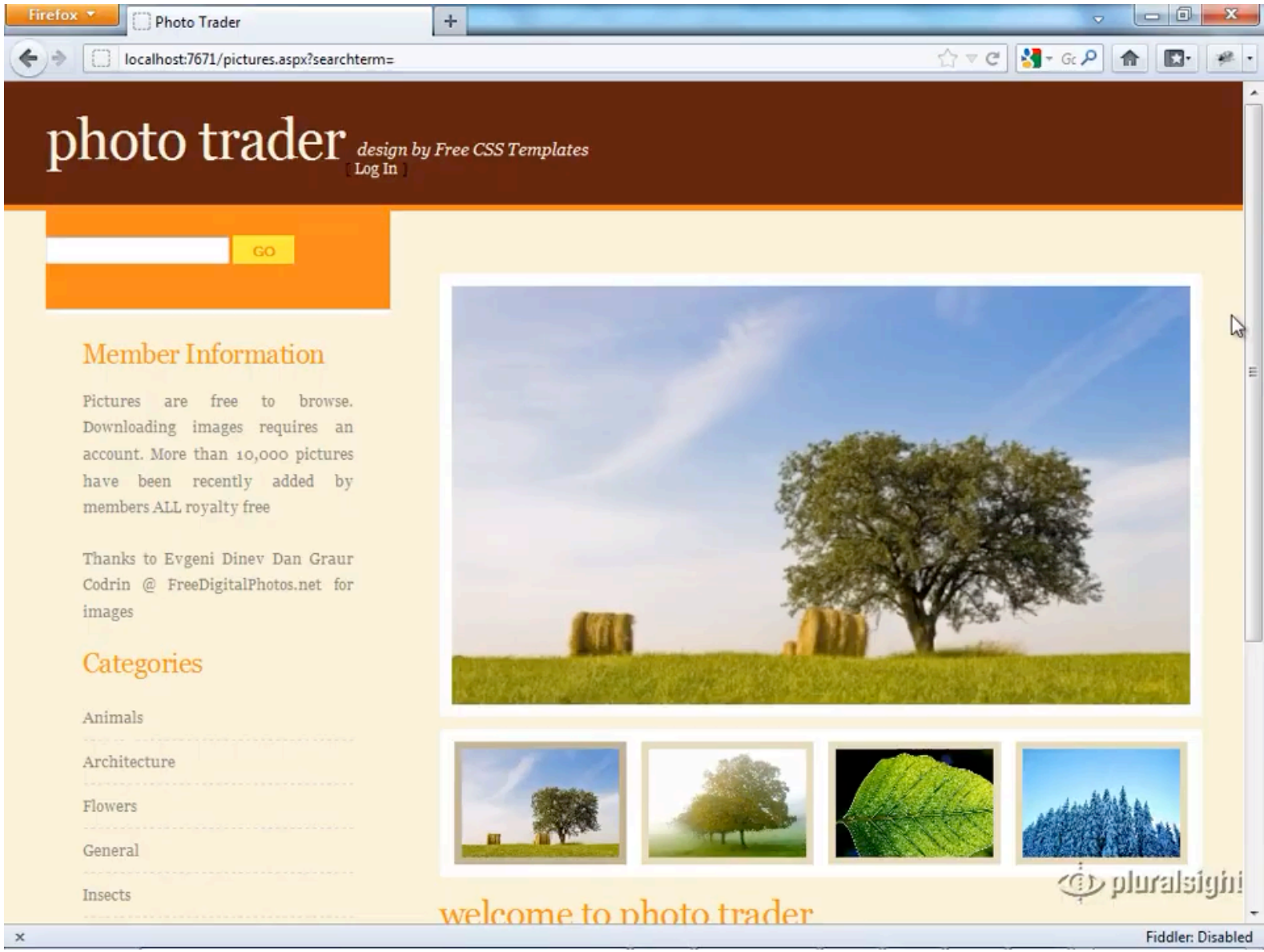
Reflected XSS



University of Victoria

© Jen

Demo


Firefox
Photo Trader
localhost:7671/pictures.aspx?searchterm=
photo trader
design by Free CSS Templates
Log In
GO
Member Information
Pictures are free to browse. Downloading images requires an account. More than 10,000 pictures have been recently added by members ALL royalty free
Thanks to Evgeni Dinev Dan Graur Codrin @ FreeDigitalPhotos.net for images
Categories
Animals
Architecture
Flowers
General
Insects
welcome to photo trader
pluralsight
Fiddler: Disabled

How to mitigate against XSS?

Never output untrusted data, except…if you carefully **encode** and sanitize it

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

this is hard!

University of Victoria

© Jens H. Weber

XSS Filter Evasions

`<IMG SRC="javascript:alert('XSS');">`	*in image tags*
`<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>`	*using reverse quotes*
`\<a onmouseover=alert(document.cookie)\>xxs link\</a\>`	*use no quotes*
`<IMG SRC="jav&#x09;ascript:alert('XSS');">`	*encode characters*
`<<SCRIPT>alert("XSS");//\<</SCRIPT>`	*unbalanced brackets*

many more test examples here: https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html

University of Victoria

© Jens H. Weber

Mutation XSS

Problem: browsers do not just display html documents, they mutate them (and they do so differently)

https://youtu.be/lG7U3fuNw3A

University of Victoria

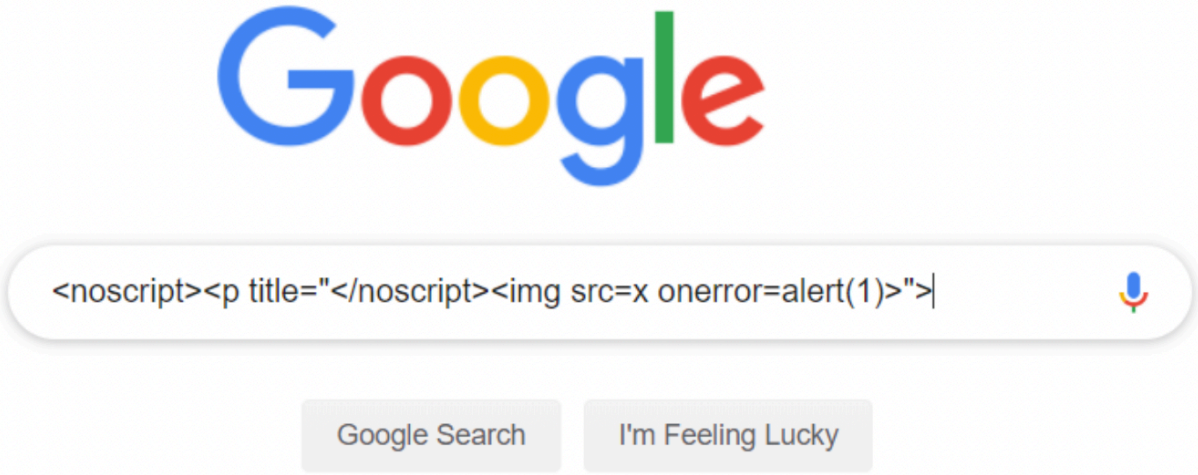
© Jens H. Weber

THE ACUNETIX BLOG › WEB SECURITY ZONE

Mutation XSS in Google Search

Tomasz Andrzej Nidecki | April 10, 2019

Are you sure that your website is safe from Cross-site Scripting if Google Search was not for five months?

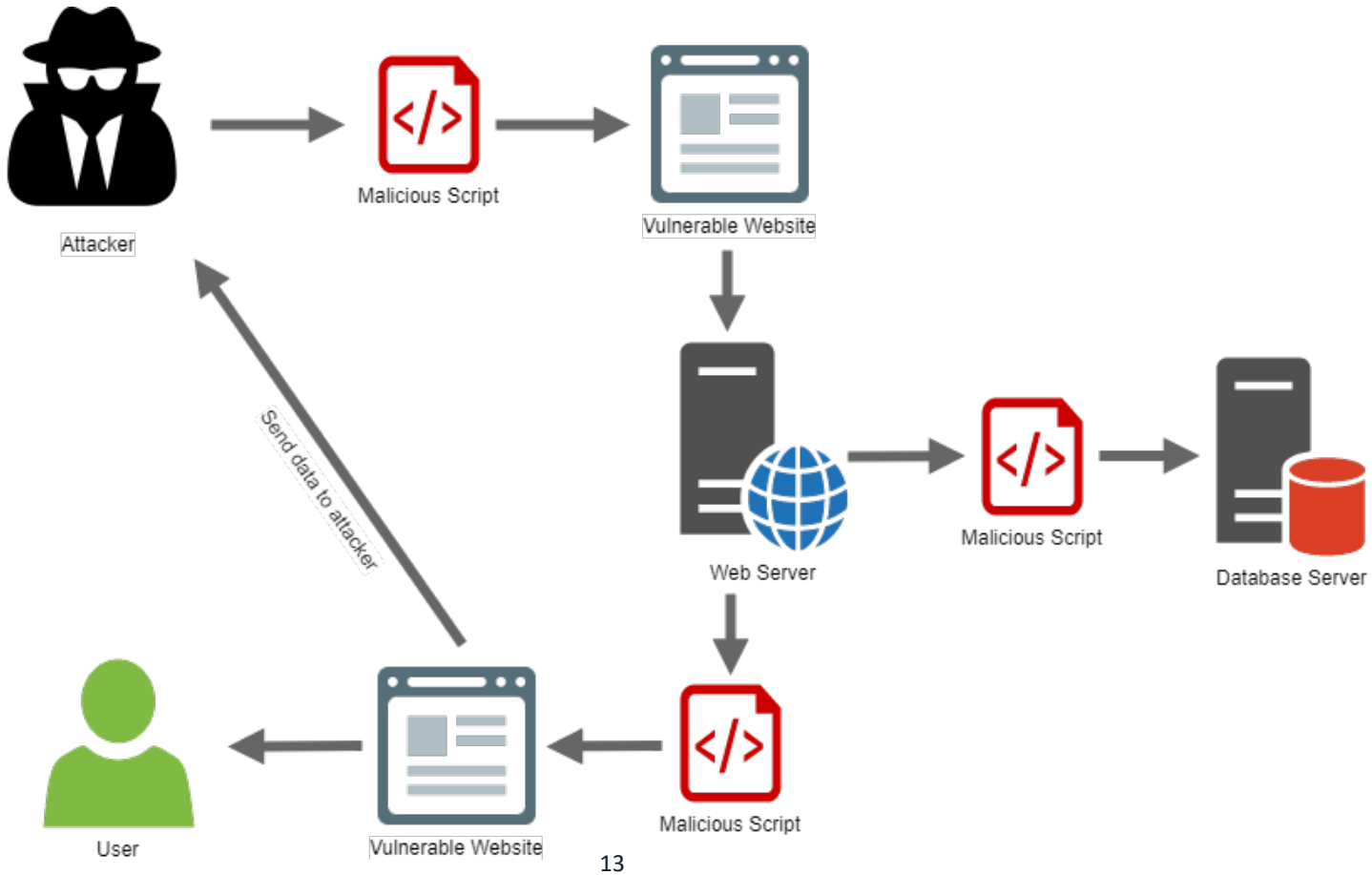


© Jens H. Weber

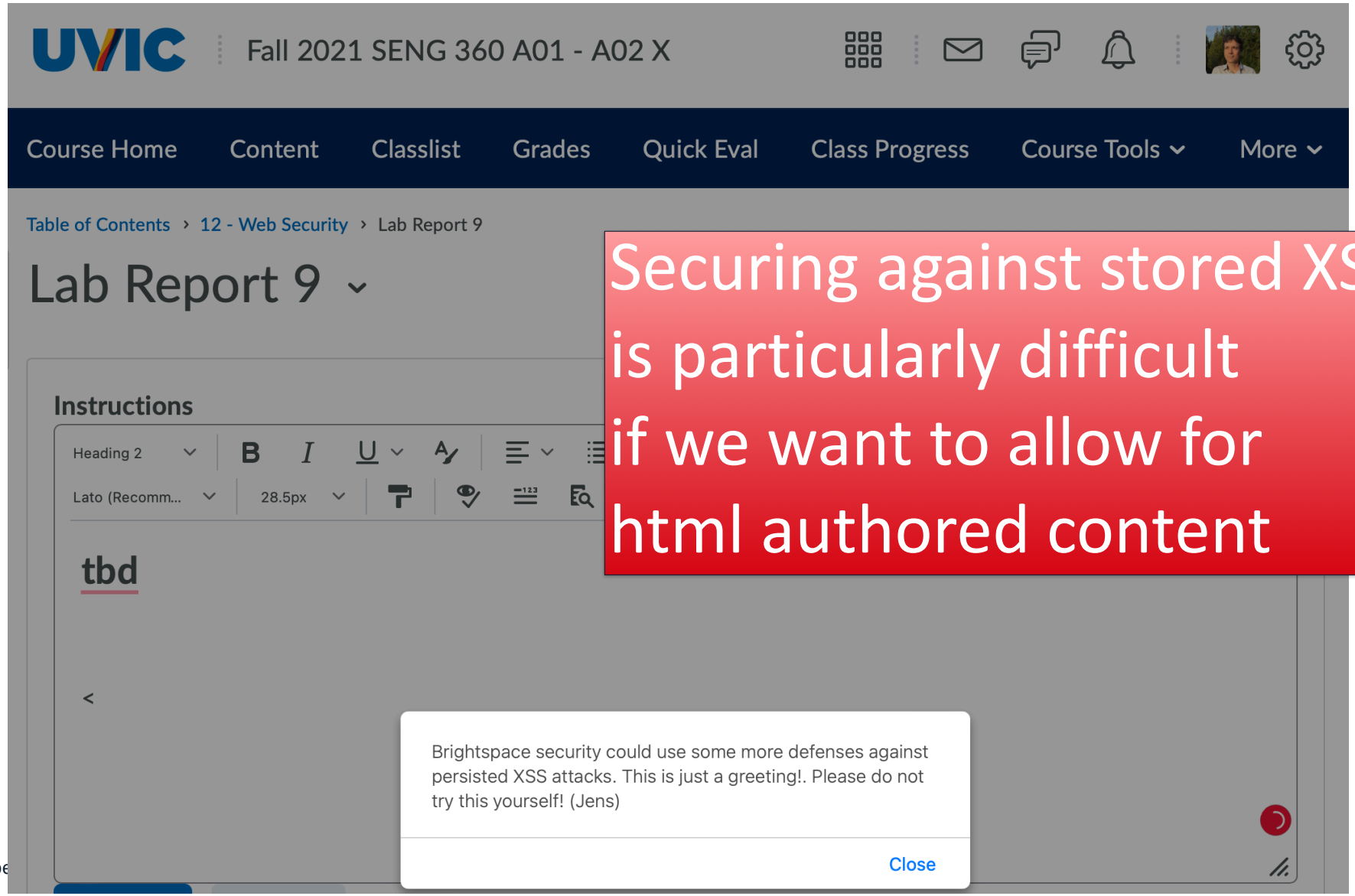
Google Search I'm Feeling Lucky

Google offered in: Deutsch

Persistent XSS



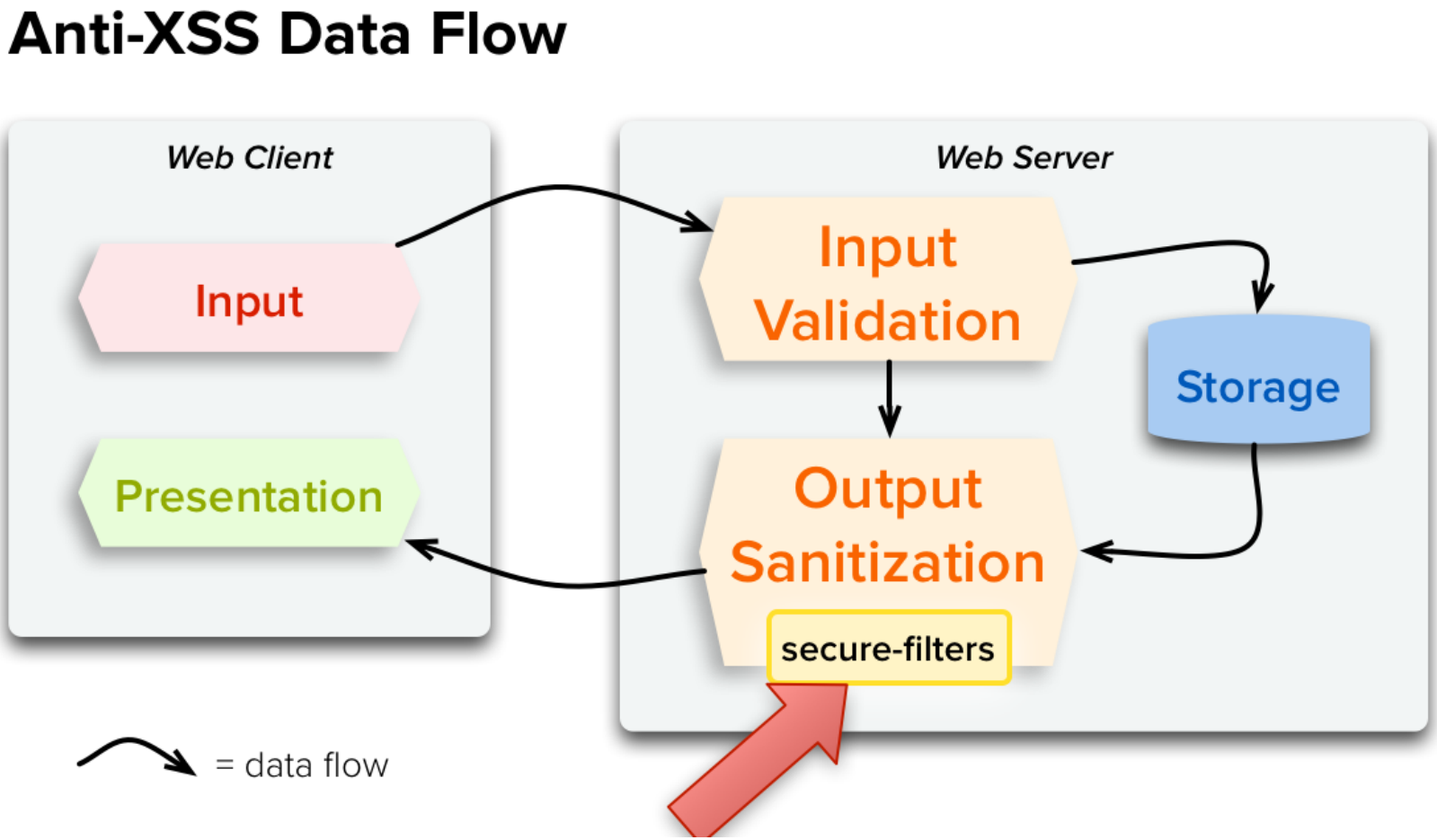
© Jens H. Weber



Securing against stored XSS is particularly difficult if we want to allow for html authored content

© Jens H. Webe

Input *and* output validation!



© Jens H. Weber

Blind XSS

Similar to stored XSS, but attacker does not know if or where injected code is stored (and if or when it will be executed).

It’s a ticking time bomb

Common target: logon forms

© Jens H. Weber

Peter H-C !OSCAR September 20, 2021 at 7:29 AM

[Oscarmcmaster-devel] Security fix login XSS vulnerability

To: oscarmcmaster-devel@lists.sourceforge.net,

Reply-To: phuttenczapski@gmail.com, oscarmcmaster-devel@lists.sourceforge.net

Versions Affected:
All OSCAR versions.

Description:
Arbitrary javascript can be entered into the schema from the login page creating a Stored Cross-Site Scripting (XSS) attack

Vulnerability test:
login with the user <script>alert('xss');</script>
open Admin > System Reports > Security Log Report
and run a login report for todays date
If vulnerable the alert will fire

when patched
1. any invalid login would be rejected and will be replaced with an error and reported in the log
2021-09-19 12:13:53.0 failed login Invalid Username
2. XSS login prior to the patch would be escaped and not fire but only displayed
2021-09-19 11:51:09.0 failed login <script>alert('xss');</script>

Risk:
Routine review of the logs by an admin user can cause arbitrary javascript to be run via stored XSS. This could include silent theft of session credentials and administrator level control.

Remediation:
Fixed in Open Oscar (thanks to Dennis for the fix)
Fixed in OSCAR oscar_emr19-40~1285.deb

© Jens

DOM-based XSS Attacks

The DOM (Domain Object Model) is the data structure to represent HTML content.

DOM-based XSS attacks use the victim's browser to reflect malicious scripts.

The scripts are *never* seen by the server

© Jens H. Weber

DOM-based XSS Attack Example

Static Web page at: **http://www.example.com/userdashboard.html?context=Mary**

```
<html>
<head>
<title>Custom Dashboard </title>
...
</head>
Main Dashboard for
<script>
        var pos=document.URL.indexOf("context=")+8;
        document.write(document.URL.substring(pos,document.URL.length));
</script>
...
</html>
```

© Jens H. Weber

Example (cont'd)

Attack:
http://www.example.com/userdashboard.html#context=<script>SomeFunction(somevariable)</script>.

```
<html>
<head>
<title>Custom Dashboard </title>
...
</head>
Main Dashboard for
<script>
        var pos=document.URL.indexOf("context=")+8;
        document.write(document.URL.substring(pos,document.URL.length));
</script>
...
</html>
```

© Jens H

https://youtu.be/ojiOCfg-FXU

Summary and Outlook

- Cross-Site Scripting (XSS) attacks exploit the fact that the client trusts the server
- Reflected, stored, blind, DOM-based
- Hard to prevent due to complexity of HTML and browsers
- Mitigations involve input and output sanitization and validation at several levels

Next class: Cross Site Request Forgery (CSERF)

University of Victoria

© Jens H. Weber

Questions?

University of Victoria

© Jens H. Weber