Security Course

WebGoat Lab sessions

Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

WebGoat Lab sessions overview

Initial Setup

- Tamper Data
- Web Goat

Lab Session 2

- HTTP Basics
- Sniffing
- Parameter Tampering

Lab Session 3

- SQL Injection
- XSS

Lab Session 4

- Access Control, session information stealing

Lab Session 5

- Authentication Flaws
- *Password cracking*

Lab Session 6

- Session Fixation/Stealing, Phishing

Why are webapplications a raising concern

- Attacks Used to be on the Operating Systems
- Now it is easier to attack the (web) applications.
- See any statistics

- Why is that so?

OLD

This page contains an overview of material for the Security (2IS05) course

OWInfo
Main page
Schedule

Below the materials for the first lectures; this page will be updated with additional materials during the course (this directory contains the currently available documents).

Lecture 1A; Introduction

Lecture notes and slides for Introduction (lecture 1A)

Suggested Reading

Book Security Engineering (First edition)
Chapter 1, Topic: General introductory text

A Goal Oriented Approach for Modeling and Analyzing Security Trade-Offs, Topic: Security requirement engineering

A metric framework to drive application security improvement, Topic: Measuring security - web page scoring

Lecture 1B-2A; Cryptography

Lecture notes and slides on Cryptography (lecture 1B-2A)

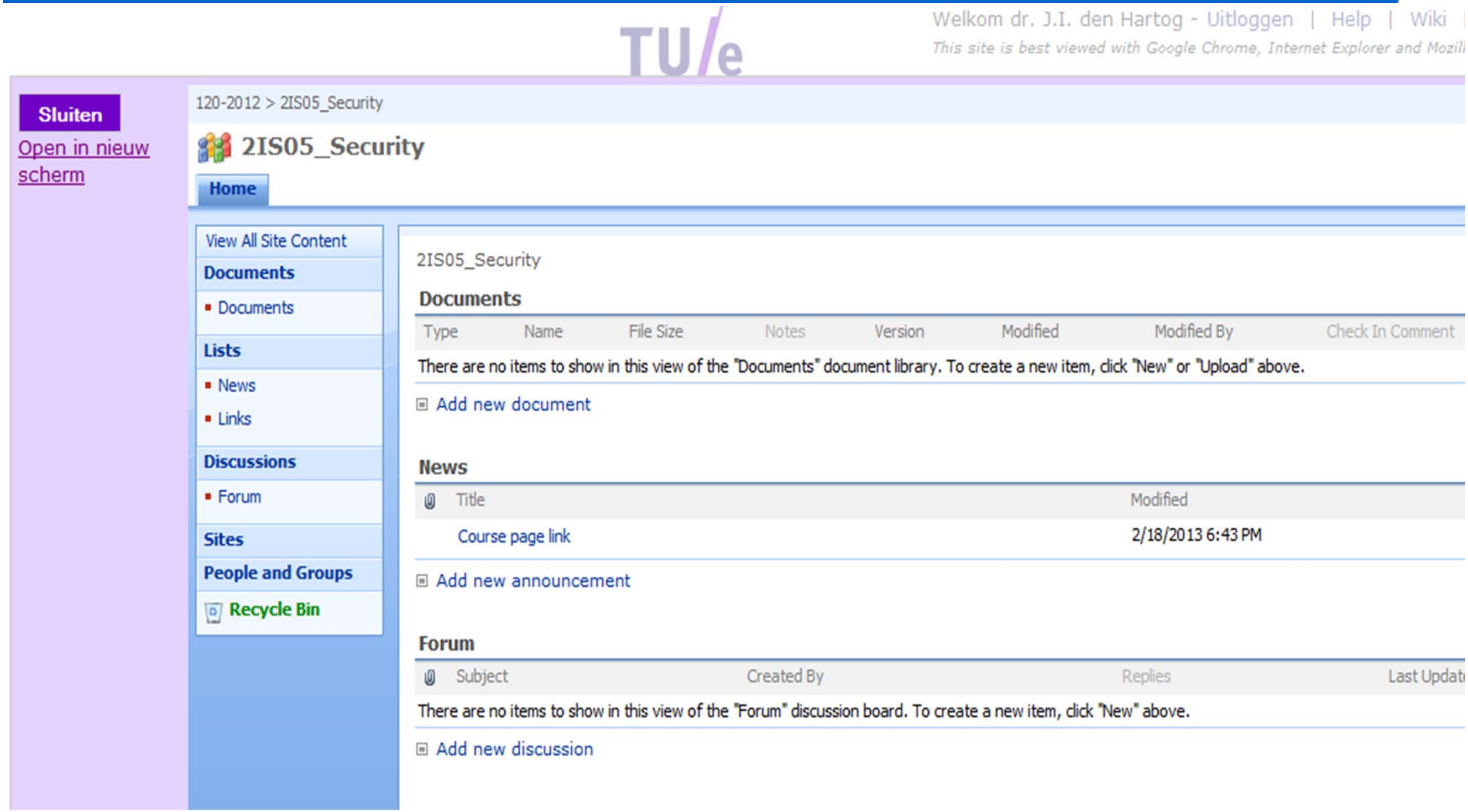
Suggested Reading

Book Security Engineering (First edition).
Chapter 5, Topic: overview cryptography

Handbook of applied cryptography, Topic: more technical treatment of cryptography
Chapter 1, Topic: overview
Chapter 7, section 1-4: (able to understand the notions but no need to know the definitions by heart for the notions not covered in class/lecture notes), Topic: block ciphers
Chapter 8, section 1, remainder of Chapter 8: able to understand working of algorithms (such as 8.2.1) (but no need to know their definitions by heart), Topic: public-key cryptography

Lecture 2B-3A; Network and web service security

New

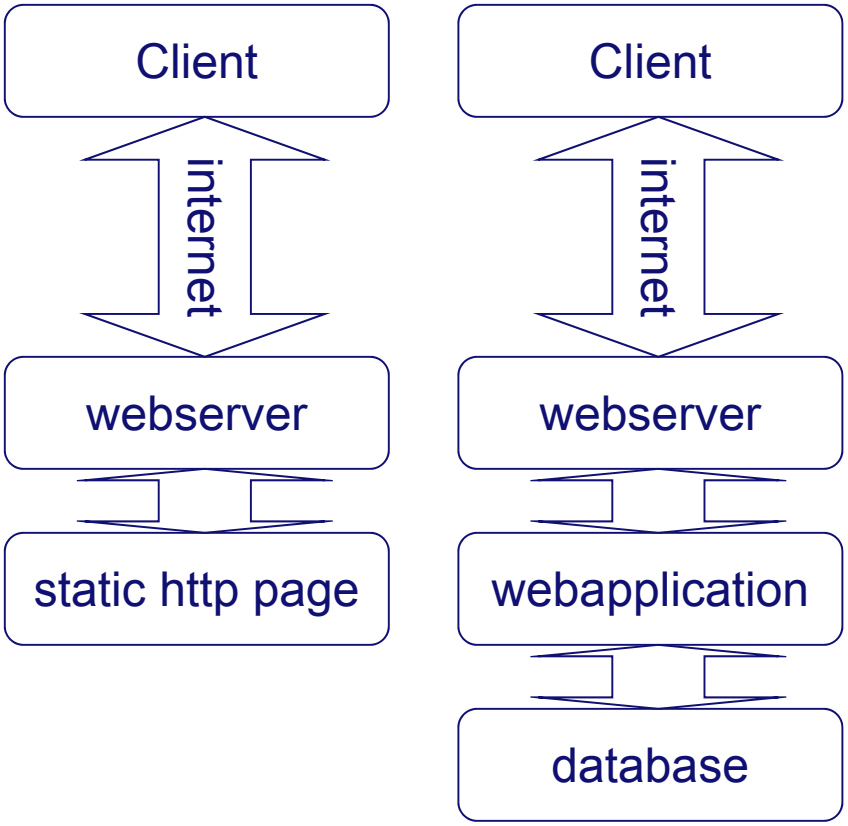


- What is the difference?

The difference

- Which parts are vulnerable?

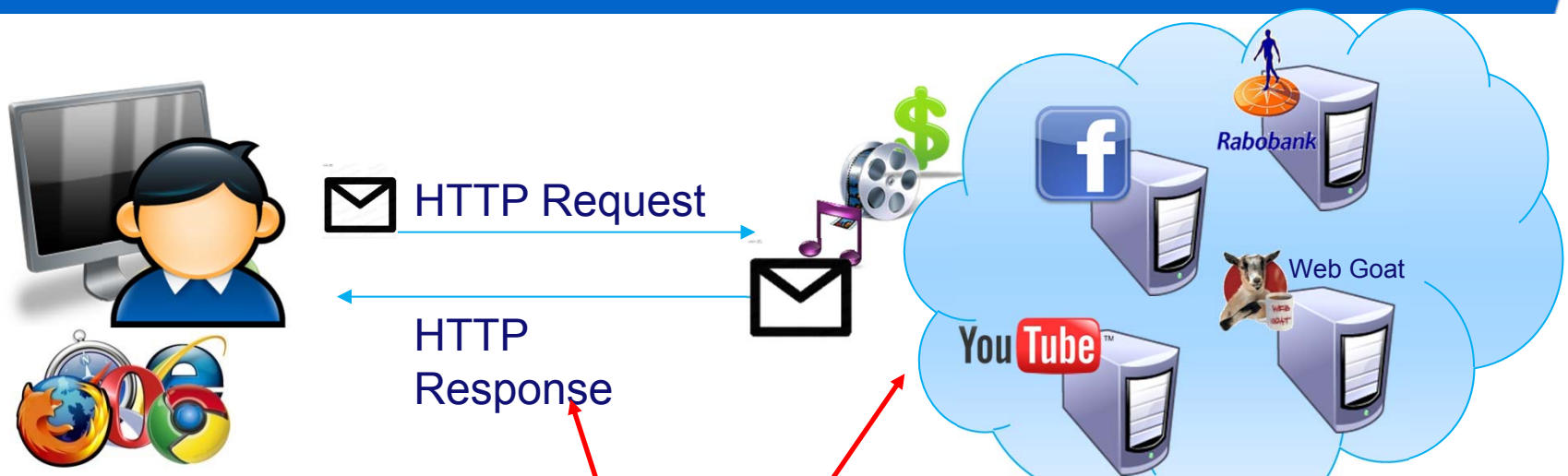
- Client:
  - Vulnerable, nothing we can do about this
- Webserver:
  - Vulnerable, but easy to harden
- Static http page:
  - Invulnerable
- WebApplication and Database:
  - Very vulnerable
  - you can have them do something for you and
  - they have access to a lot of information (usernames, passwords…)



WEB APPLICATION ARCHITECTURE

TU/e Technische Universiteit Eindhoven University of Technology

Web Application Architecture



User

Sends requests to the services he wants to use (e.g. Facebook, Google, YouTube)

Web server

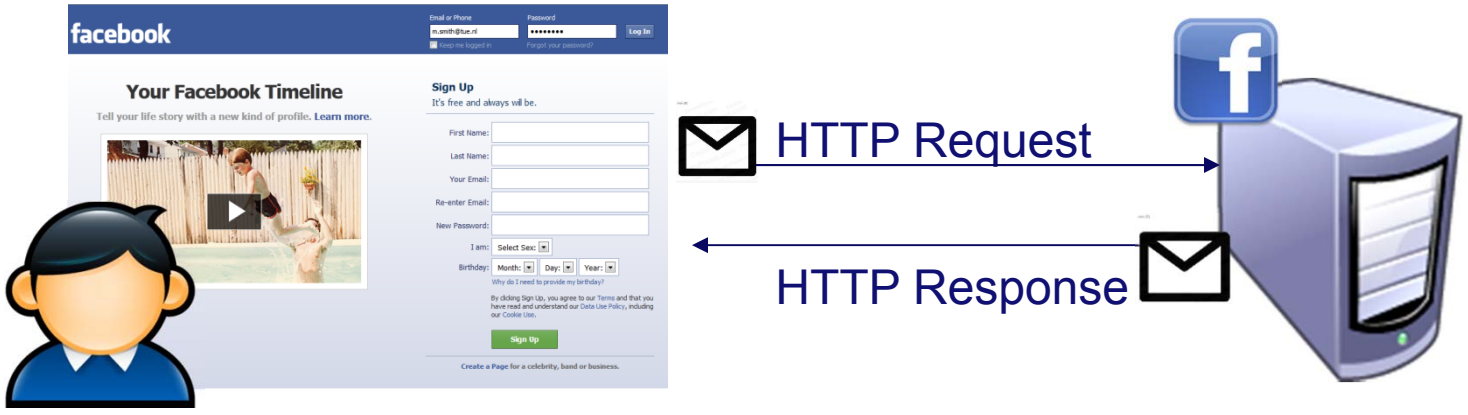
Web servers listen for users requests and sends the response (either he wants to listen a song, or visit a friend's profile)

Intruder

- Can play the role of the user, and modify the HTTP request and response
- Can access directly the web servers to exploit vulnerabilities

HTTP Request/Response

- While browsing, every time an action is taken, a **HTTP Request** is created
- The **HTTP Request** goes from the browser to the web server
- The web server make some elaboration (e.g. verify if you are a registered user) and send back a **HTTP Response**



HTTP Request

Your Facebook Timeline

Sign Up

It's free and always will be.

HTTP REQUEST

Tamper Popup

https://www.facebook.com/login.php?login_attempt=1

Request Header Name	Request Header Value
Host	www.facebook.com
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64;
Accept	text/html,application/xhtml+xml,applic
Accept-Language	en-us,en;q=0.5
Accept-Encoding	gzip, deflate
Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection	keep-alive
Referer	http://www.facebook.com/
Cookie	datr=Md3hT0bgI-La0GG3BwWQVHFy; l

Post Parameter Name	Post Parameter Value
lsd	AVo5wG5I
email	m.smith%40tue.nl
pass	12345678
default_persistent	0
charset_test	%E2%82%AC%2C%C2%B4%2C%E2%82
timezone	-120
lgnrnd	072625_JTCT
lgnjs	1340202386
locale	en_US

OK Cancel

username and *password* are sent (most likely in clear!!) over the network. They can be intercepted

TU/e Technische Universiteit Eindhoven University of Technology

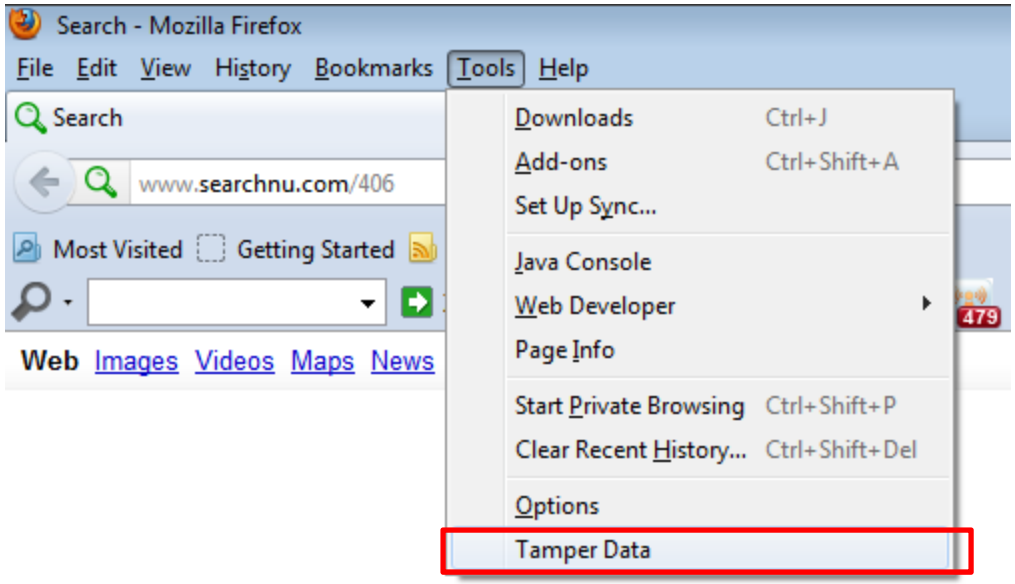
HTTP Response

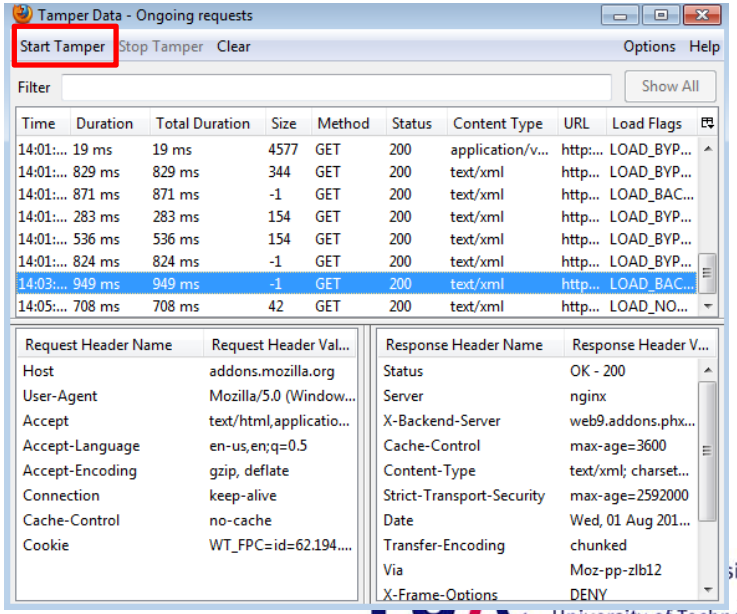
Response Header Name	Response Header Value
Status	OK - 200
Cache-Control	private, no-cache, no-store, must-revalidate
Expires	Sat, 01 Jan 2000 00:00:00 GMT
P3P	CP="Facebook does not have a P3P policy. Learn why here: http://fb.me/p3p"
Pragma	no-cache
X-Content-Type-Options	nosniff
X-Frame-Options	DENY
Set-Cookie	datr=Md3hT0bgI-La0GG3BwWQVHFy; expires=Fri, 20-Jun-2014 14:54:05 GMT; path=/; domain=.facebook.com; httponlywd=...
Content-Encoding	gzip
Content-Type	text/html; charset=utf-8
X-FB-Debug	51HwMOf/IHVsaNhbdlBfahcig+JwacTtsObqGGSPI8c=
Date	Wed, 20 Jun 2012 14:54:05 GMT
Transfer-Encoding	chunked
Connection	keep-alive

INITIAL SETUP

Tamper Data

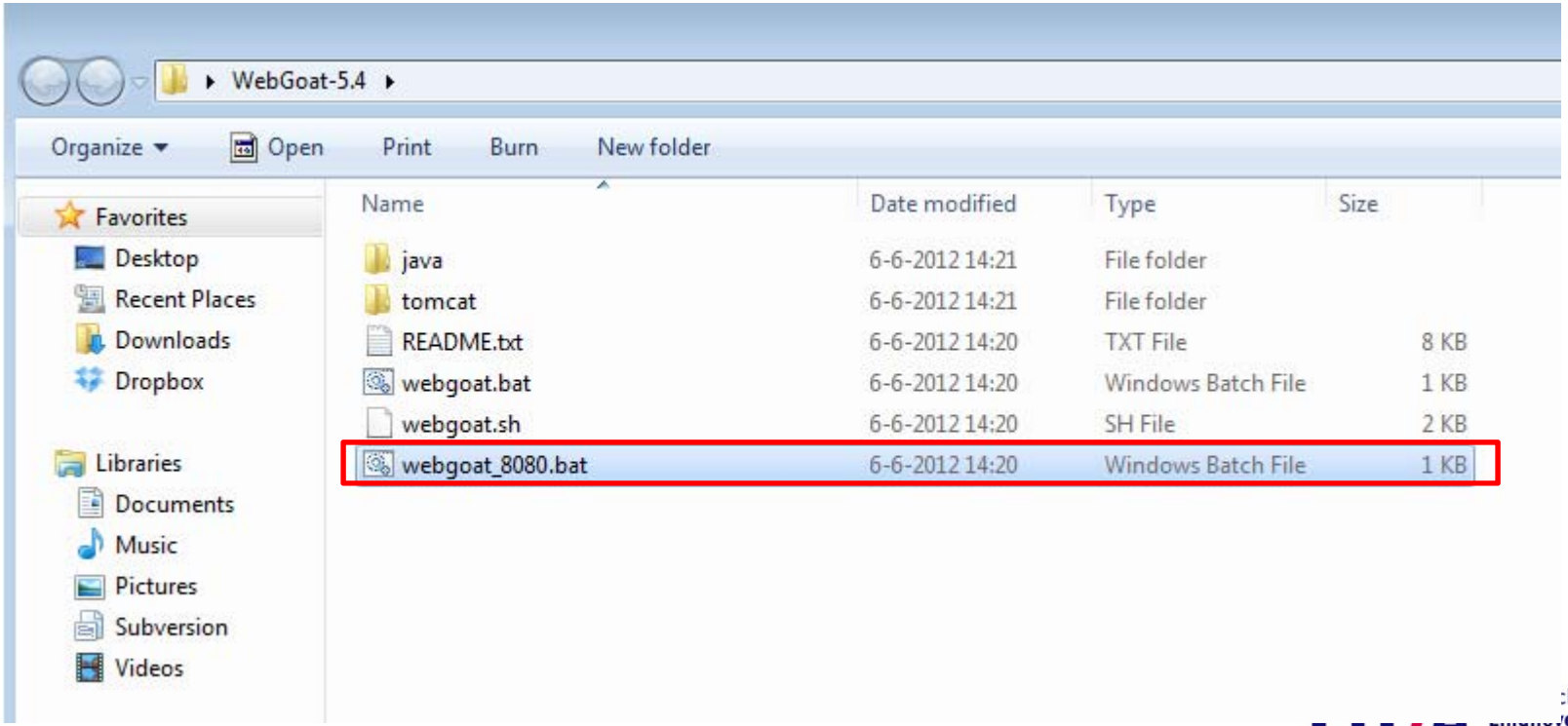
- Tamper Data is a tool allowing you to intercept and modify Request/Response from your Mozilla Firefox Browser
- If not yet installed, you can download it here: https://addons.mozilla.org/en-us/firefox/addon/tamper-data/
- You have to click on "*Start Tamper*" to start intercepting Request/Response
- Note that this will intercept, and let you see the HTTP request/response, all your internet traffic
  - you have to "*Stop Tamper*" to get back to normal browsing





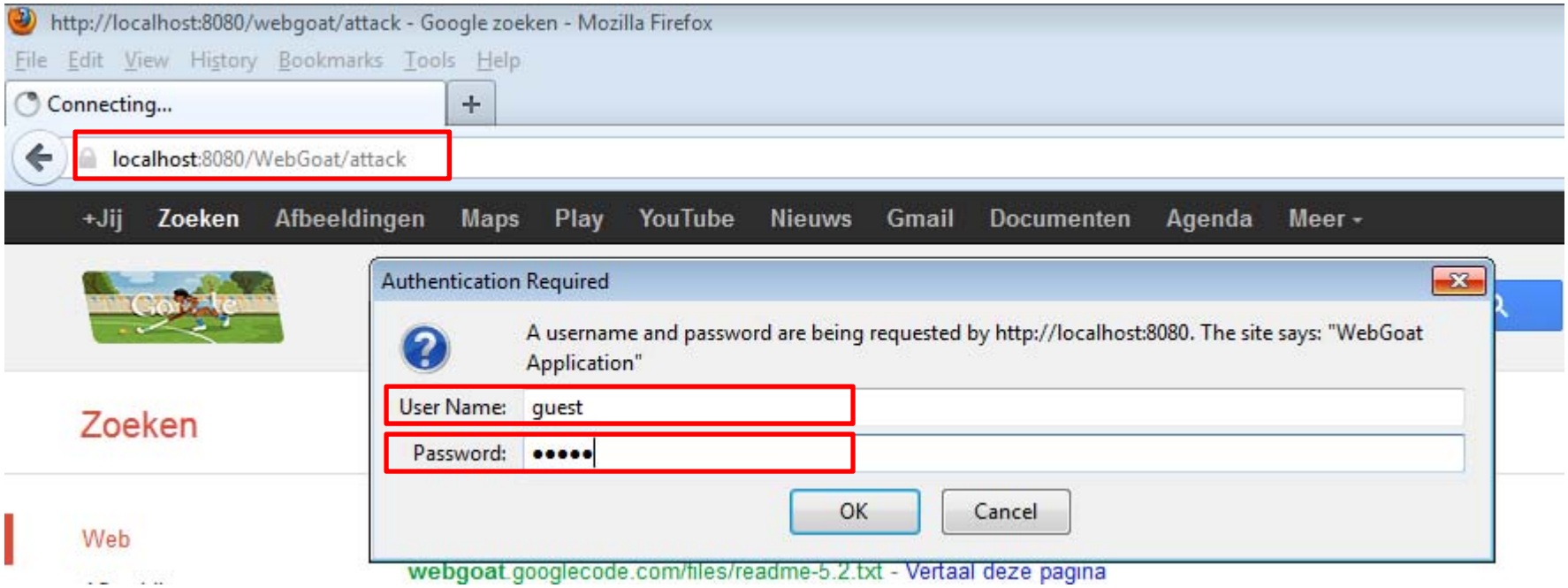
WEB GOAT (1)

- Close your Internet Connection (your machine is **extremely** vulnerable when **WebGoat** is running)
- Go to the folder containing your WebGoat installation
- Execute the **webgoat_8080.bat** file



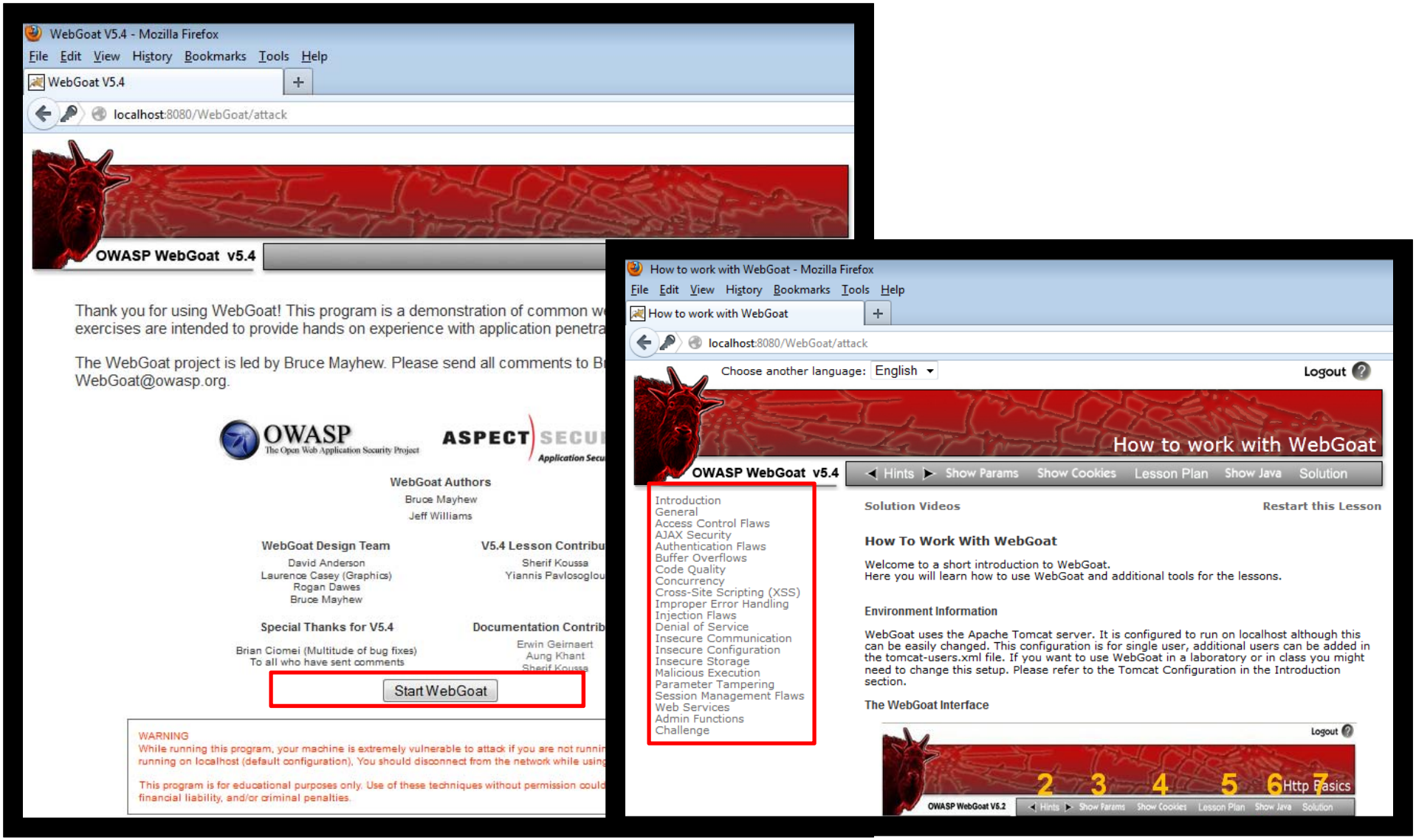
WEB GOAT (2)

- **Type the address http://localhost:8080/WebGoat/attack in Mozilla Firefox**
- **Login as username = guest and pwd= guest**



WEB GOAT – Setup

- **Press “Start WebGoat” to access the *Lesson Section***



LAB SESSION 2

TU/e Technische Universiteit Eindhoven University of Technology

Where to find exercises in WebGoat

- **Lab Session 2**

  HTTP Basics:
  - General
    - HTTP Basics

  Sniffing:
  - Insecure Communication
    - Insecure login

  Parameter Tampering:
  - Parameter Tampering
    - Bypass HTML Field Restrictions
      - Exploit Hidden Fields

- ***Lab Session 3***

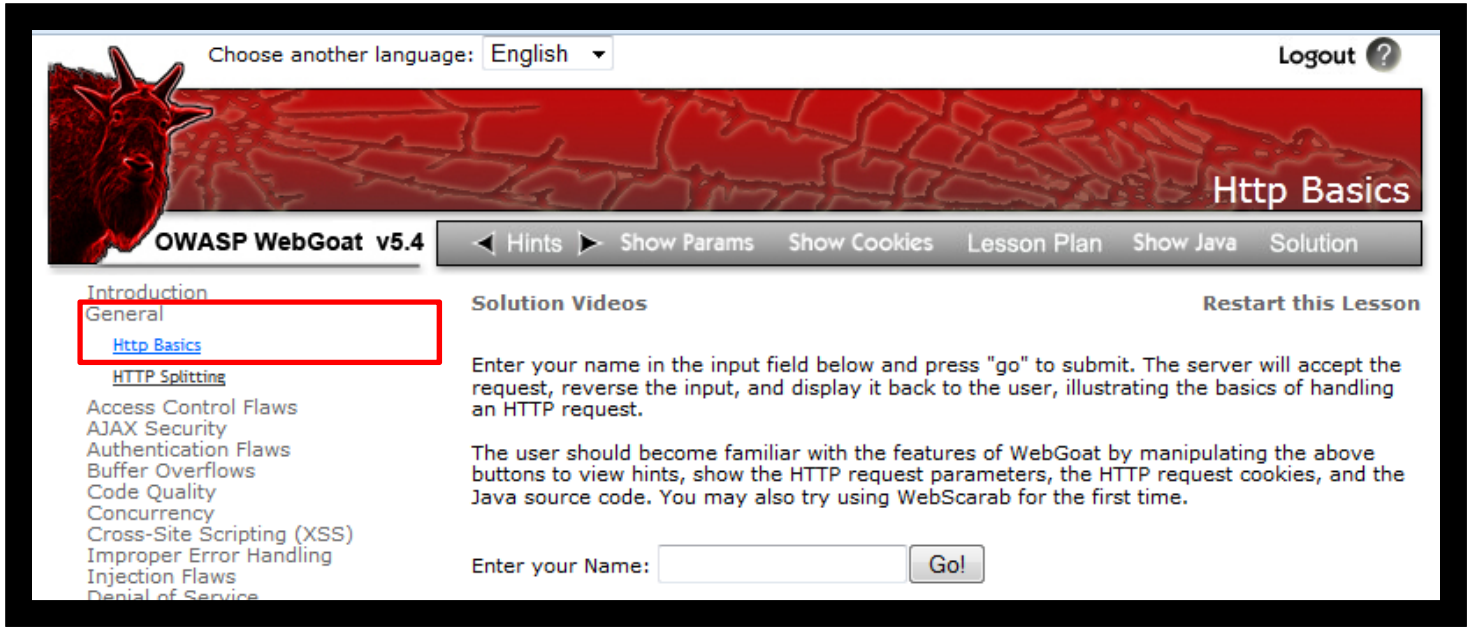
  SQL Injection
  - *Injection Flaw*
    - *Modify data with SQL injection*

  XSS
  - *Xross-Site-Scripting (XSS)*
    - *Stage 1: Stored XSS*

- ***Lab Session 4***

  Access Control
  - *Access Control*
    - *Stage 3: Bypass Data Layer Access Control*

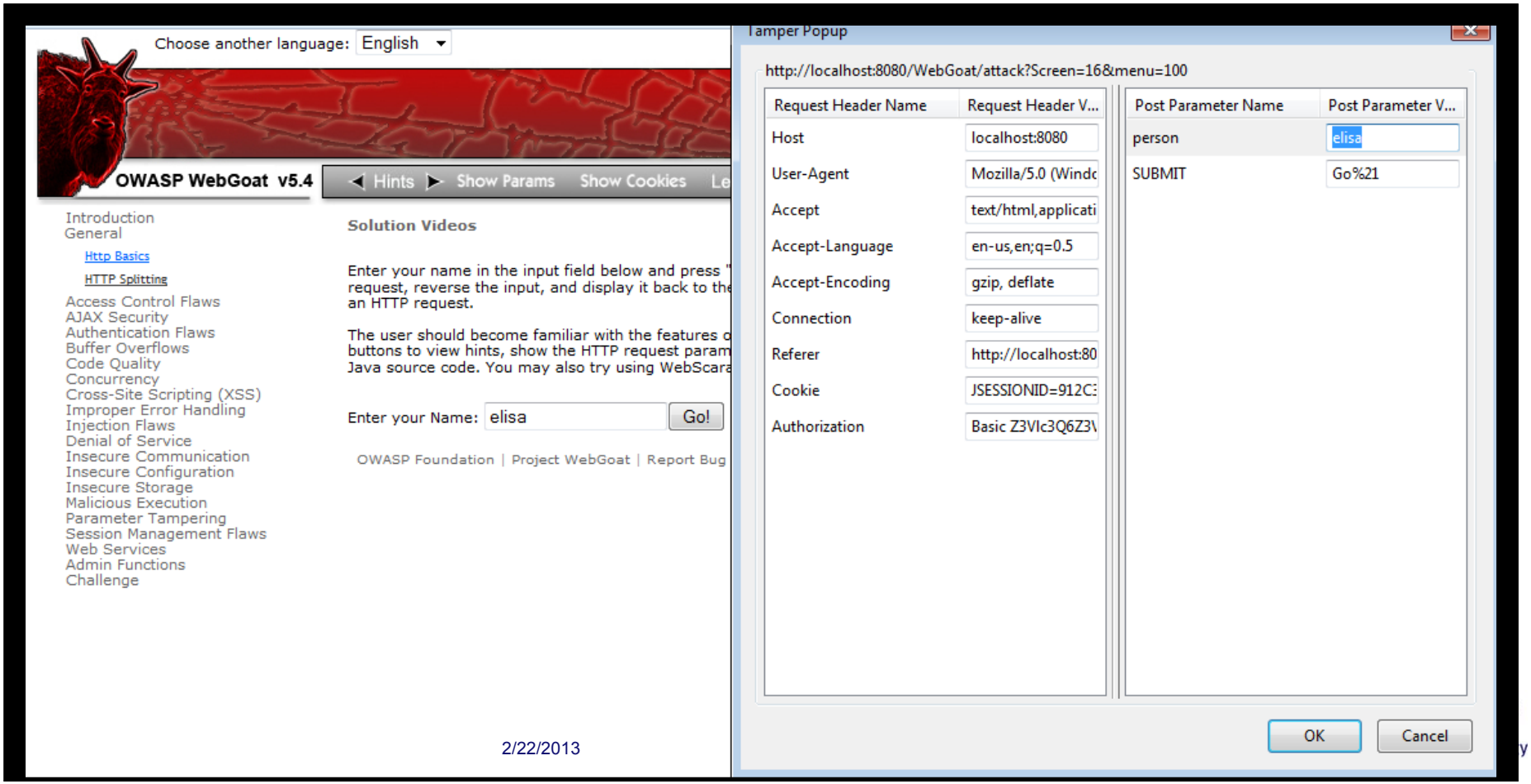
HTTP Basics - Exercise



- *Goal*: meet WebGoat and TamperData.
- *Exercise:*
  - *Go to;* exercise **General → Http Basics**
  - Insert your name in the input field and start the tampering
  - Modify the parameter '*person*' in the HTTP request in such a way to get back the string "webgoat" as response from the server

HTTP Basics - Solution

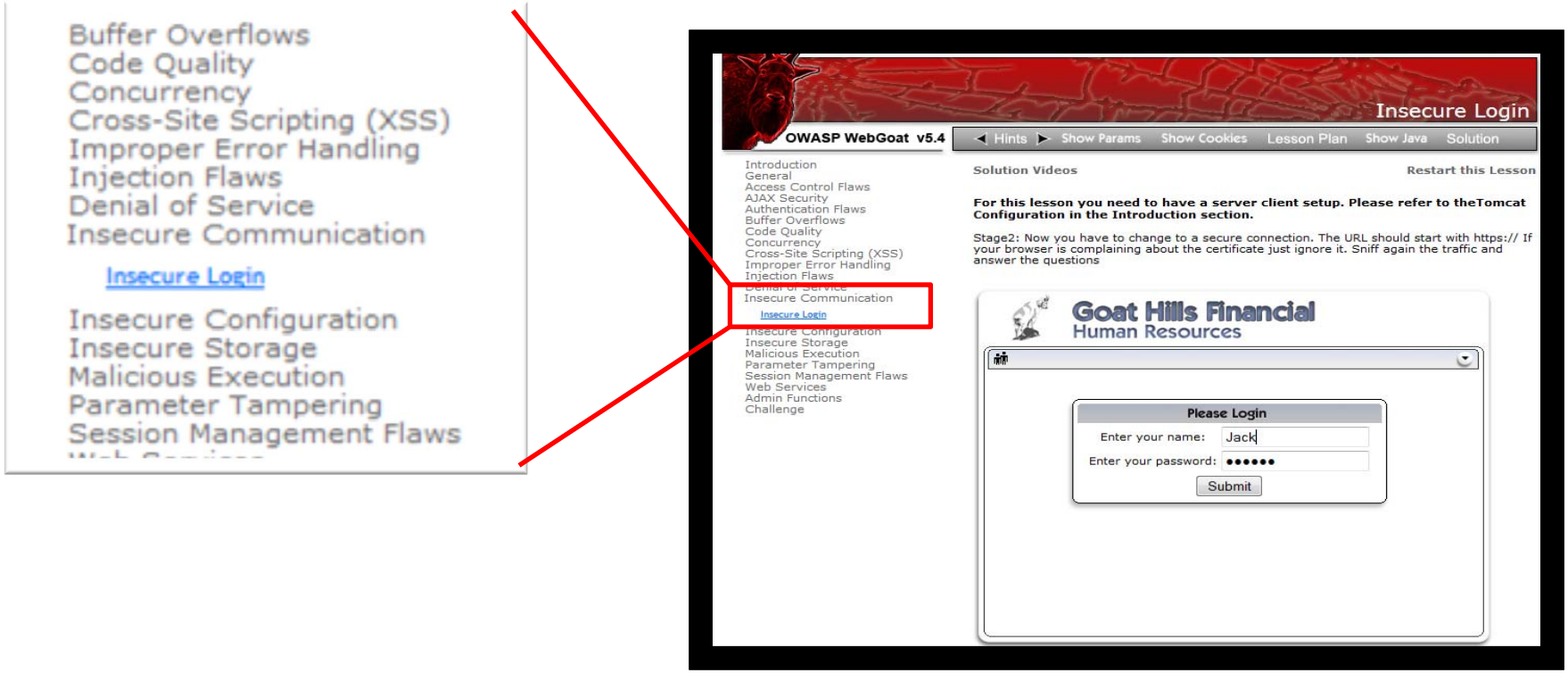
- Change the value of ‘person’ to *taogbew*
- The server will reverse it and you will get “webgoat” as final response.



HTTP Basics - Lesson learned

- When parameters are in clear (i.e. not encrypted) they can be easily changed by who is listening your internet traffic.
  - In this case it was ***only*** your name
  - But…
  - Assume you want to make a payment of 800 Euro to the account of your landlord and insert **12345** as the account number
  - The attacker can change such number to **34566** (his account number)
  - In this way he managed to steal 800 Euro from you

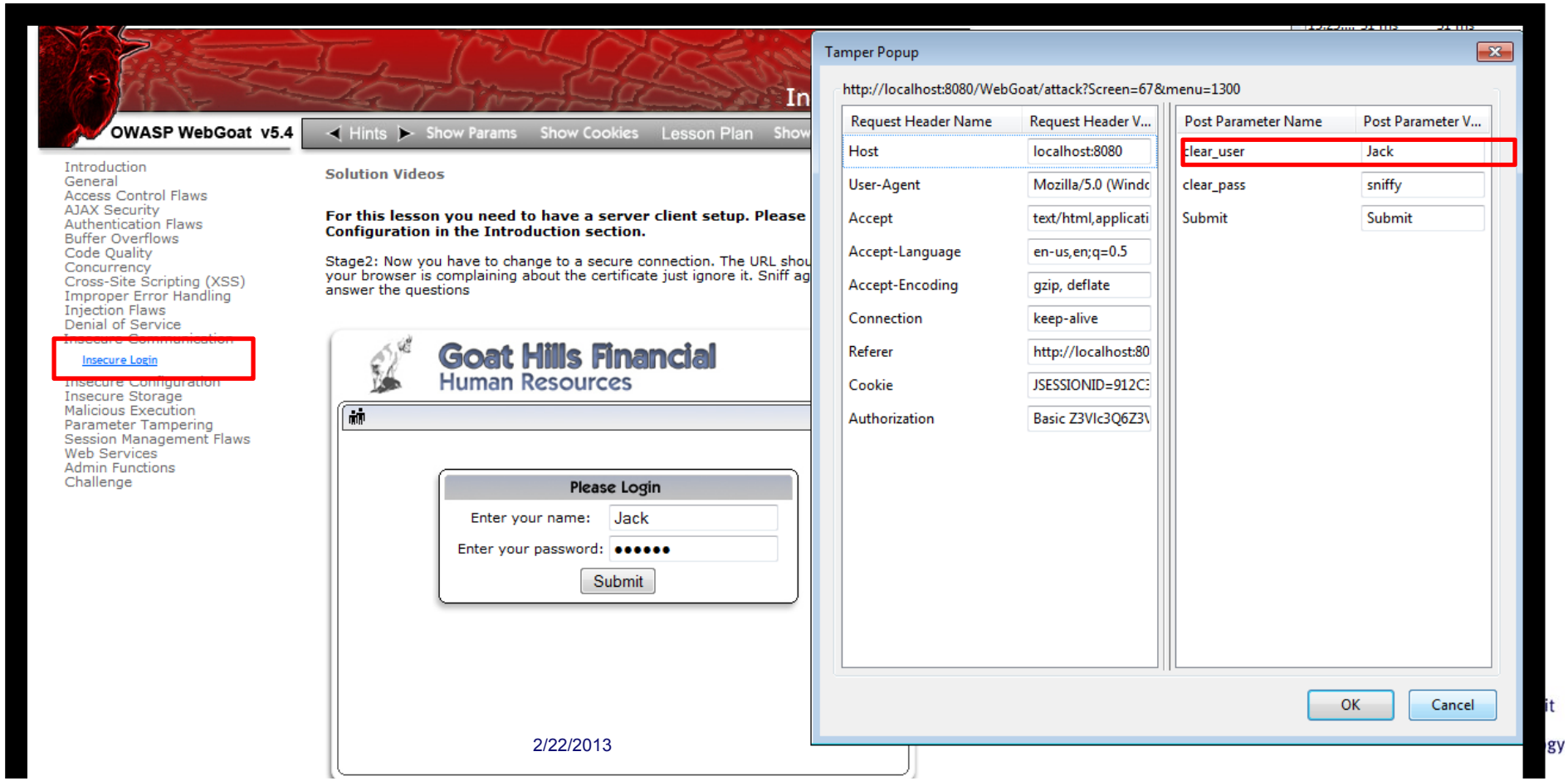
Sniffing - Exercise



- *Goal:* Steal the password of the user *Jack*
- *Exercise:*
  - Go to **Insecure Communication → Insecure Login**
  - Press the button *Submit* and use *Tamper Data* to steal the password

Sniffing - Solution

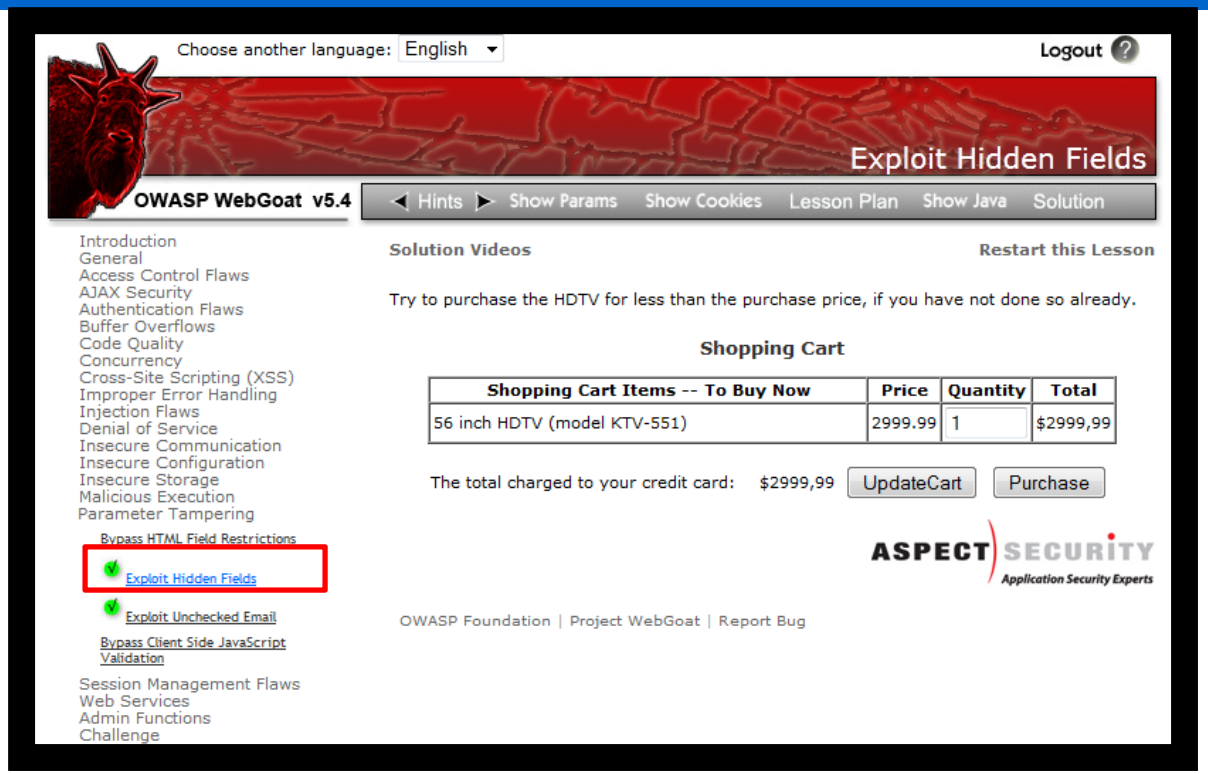
- Start tampering then press the *Submit* button
- Get the value of the field *clear_pass*
- The solution is *“sniffy”*



Sniffing - Lesson learned

- You performed your first sniffing attack
- You intercepted the traffic of your victim and stolen his password
- If this is the same password he uses for his internet banking (or email account) you can now easily access it

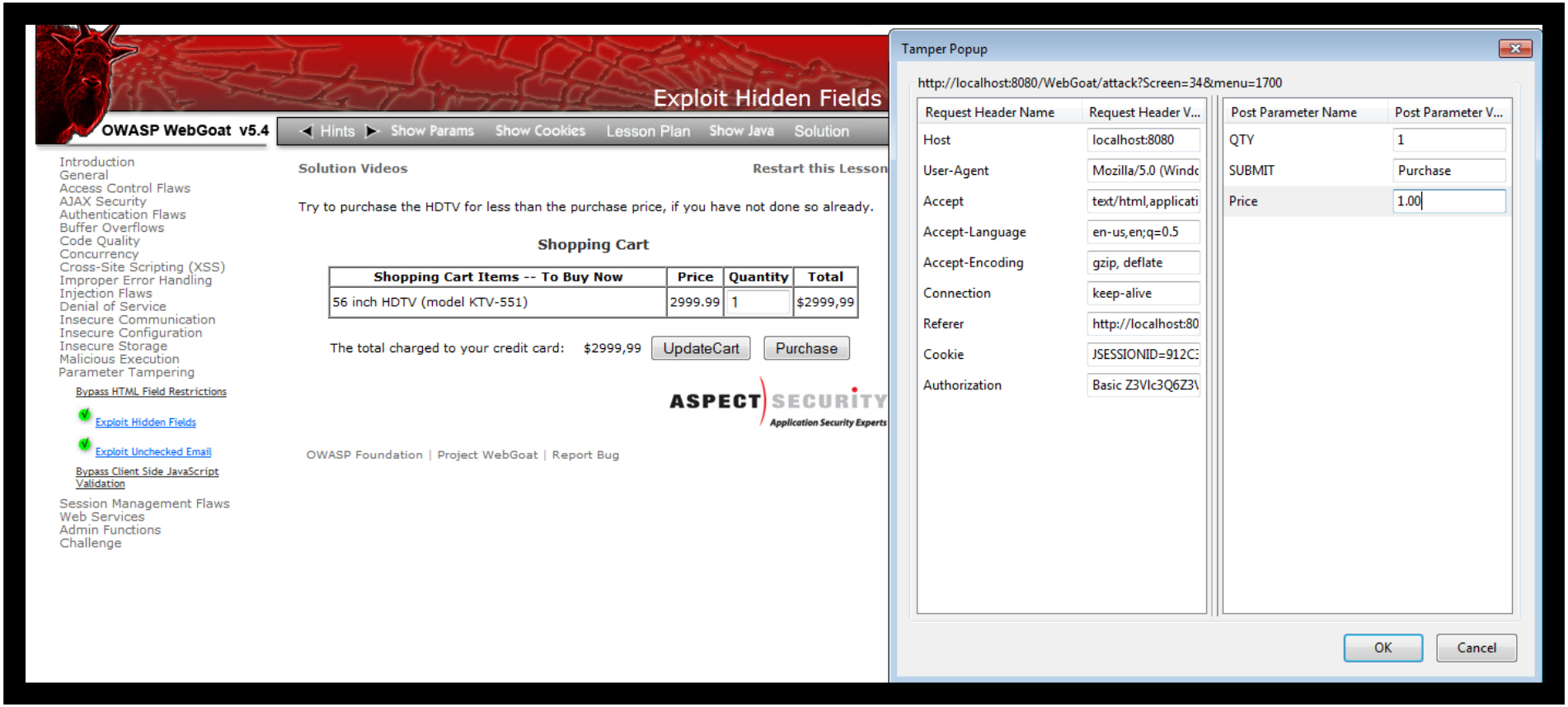
Parameter Tampering – Exercise



- *Goal:* change the total amount charged to your credit card
- *Exercise:*
  - Go to **Parameter Tampering→ Exploit Hidden Fields**
  - Purchase the TV for 1$

Parameter Tampering – Solution

- Start Tampering Data then press the button *Purchase*
- Change the parameter *Price* to the value 1.00$
- If successful you will get a *Congratulations* message



Lesson learned

- You used your recently learned “hacking” skills to gain personal advantages
  - You paid 1$ a product worth 3000$
- Why is that possible?
  - The web server is not checking that you’re paying the right amount of money
  - An hacker who knows this vulnerability is able to exploit it