

Security Course

WebGoat Lab sessions



TU / **e** 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

Sluiten
[Open in nieuw scherm](#)

120-2012 > 2IS05_Security

2IS05_Security

[Home](#)

- View All Site Content
- Documents**
 - Documents
- Lists**
 - News
 - Links
- Discussions**
 - Forum
- Sites**
- People and Groups**
- [Recycle Bin](#)

2IS05_Security Documents

Type	Name	File Size	Notes	Version	Modified	Modified By	Check In Comment
There are no items to show in this view of the "Documents" document library. To create a new item, click "New" or "Upload" above.							
<input type="checkbox"/> Add new document							

2IS05_Security News

@	Title	Modified
	Course page link	2/18/2013 6:43 PM
<input type="checkbox"/> Add new announcement		

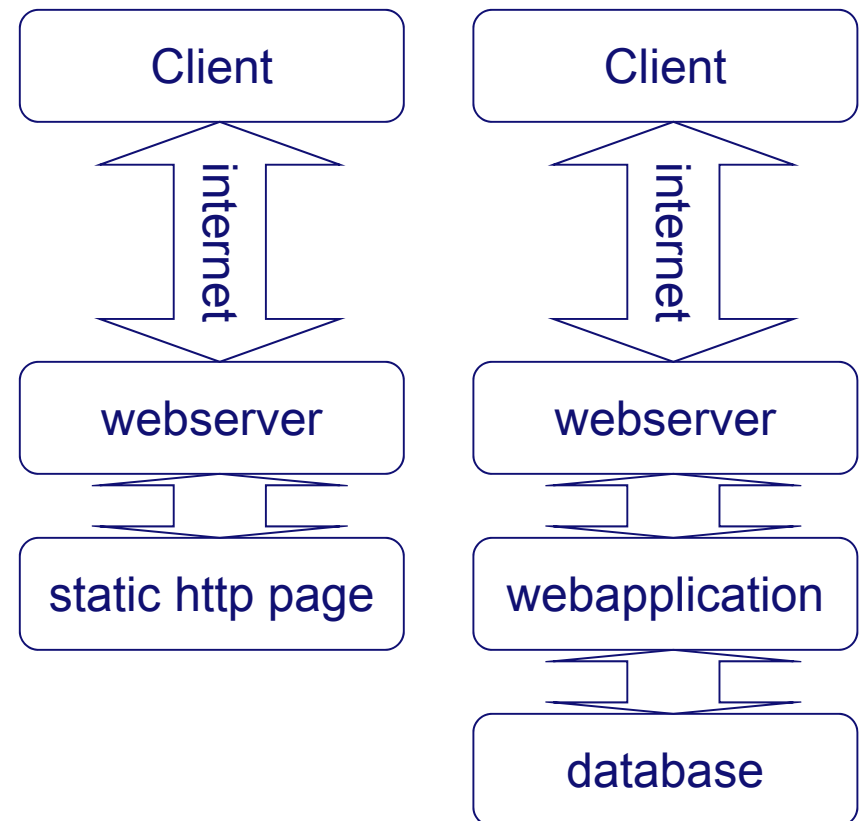
2IS05_Security Forum

@	Subject	Created By	Replies	Last Update
There are no items to show in this view of the "Forum" discussion board. To create a new item, click "New" above.				
<input type="checkbox"/> Add new discussion				

- What is the difference?

The difference

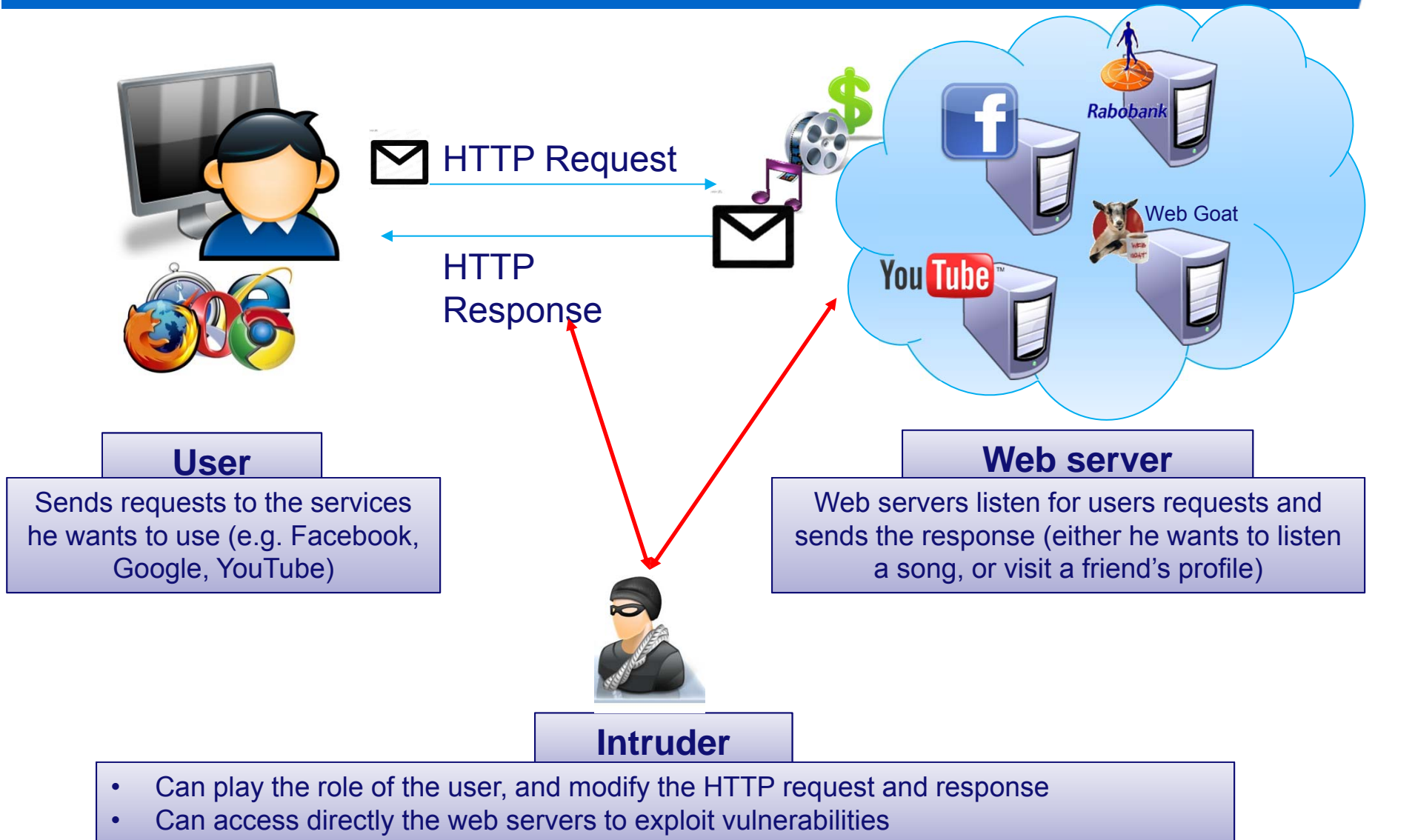
- Which parts are vulnerable?
- Client:
 - Vulnerable, nothing we can do about this
- Webserver:
 - Vulnerable, but easy to harden
- Static http page:
 - Invulnerable
- WebApplication and Database:
 - Very vulnerable
 - you can have them do something for you and
 - they have access to a lot of information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





WEB APPLICATION ARCHITECTURE

Web Application Architecture



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

facebook

Email or Phone

m.smith@tue.nl

Password

.....

Log In

Keep me logged in

[Forgot your password?](#)

Your Facebook Timeline

Sign Up

It's free and always will be.

Tamper Popup

HTTP REQUEST

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

Request Header Name	Request Header Value	Post Parameter Name	Post Parameter Value
Host	www.facebook.com	Isd	AVo5wG5I
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64; ...)	email	m.smith%40tue.nl
Accept	text/html,application/xhtml+xml,application/javascript;q=0.9,*/*;q=0.8	pass	12345678
Accept-Language	en-us,en;q=0.5	default_persistent	0
Accept-Encoding	gzip, deflate	charset_test	%E2%82%AC%2C%C2%B4%2C%E2%82%A4
Accept-Charset	ISO-8859-1,utf-8;q=0.7,*;q=0.7	timezone	-120
Connection	keep-alive	Ignrnd	072625_JTCT
Referer	http://www.facebook.com/	Ignjs	1340202386
Cookie	datr=Md3hT0bgI-La0GG3BwWQVHFy; l...	locale	en_US

OK Cancel

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



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/IHVsaNhbdIBfahcig+JwacTtsObqGGSPi8c=
Date	Wed, 20 Jun 2012 14:54:05 GMT
Transfer-Encoding	chunked
Connection	keep-alive

The response can be intercepted and parameters values can be changed

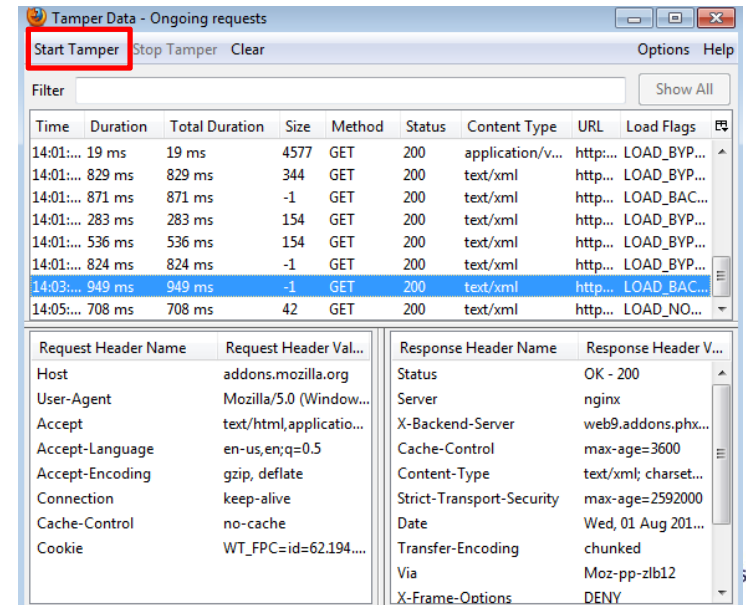
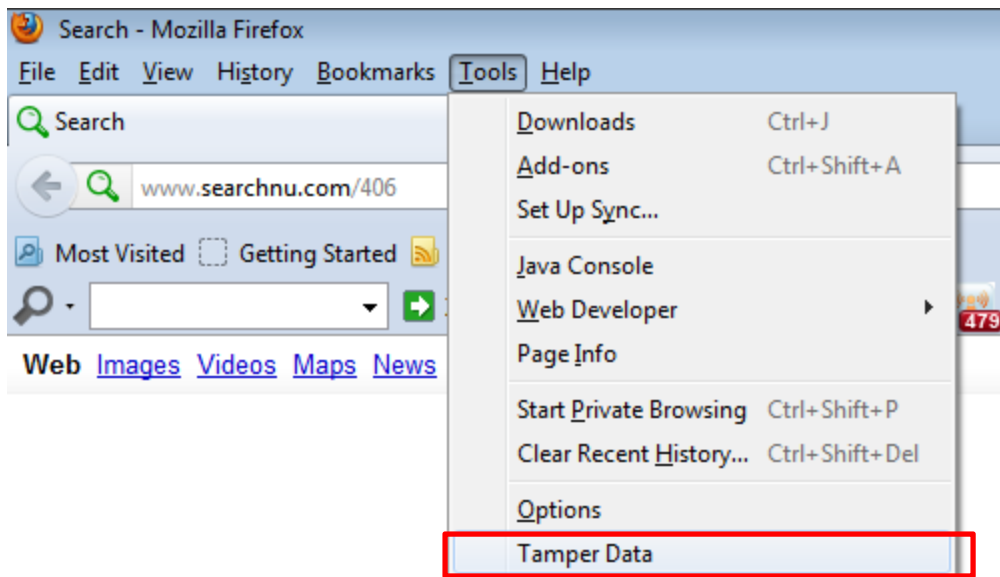




INITIAL SETUP

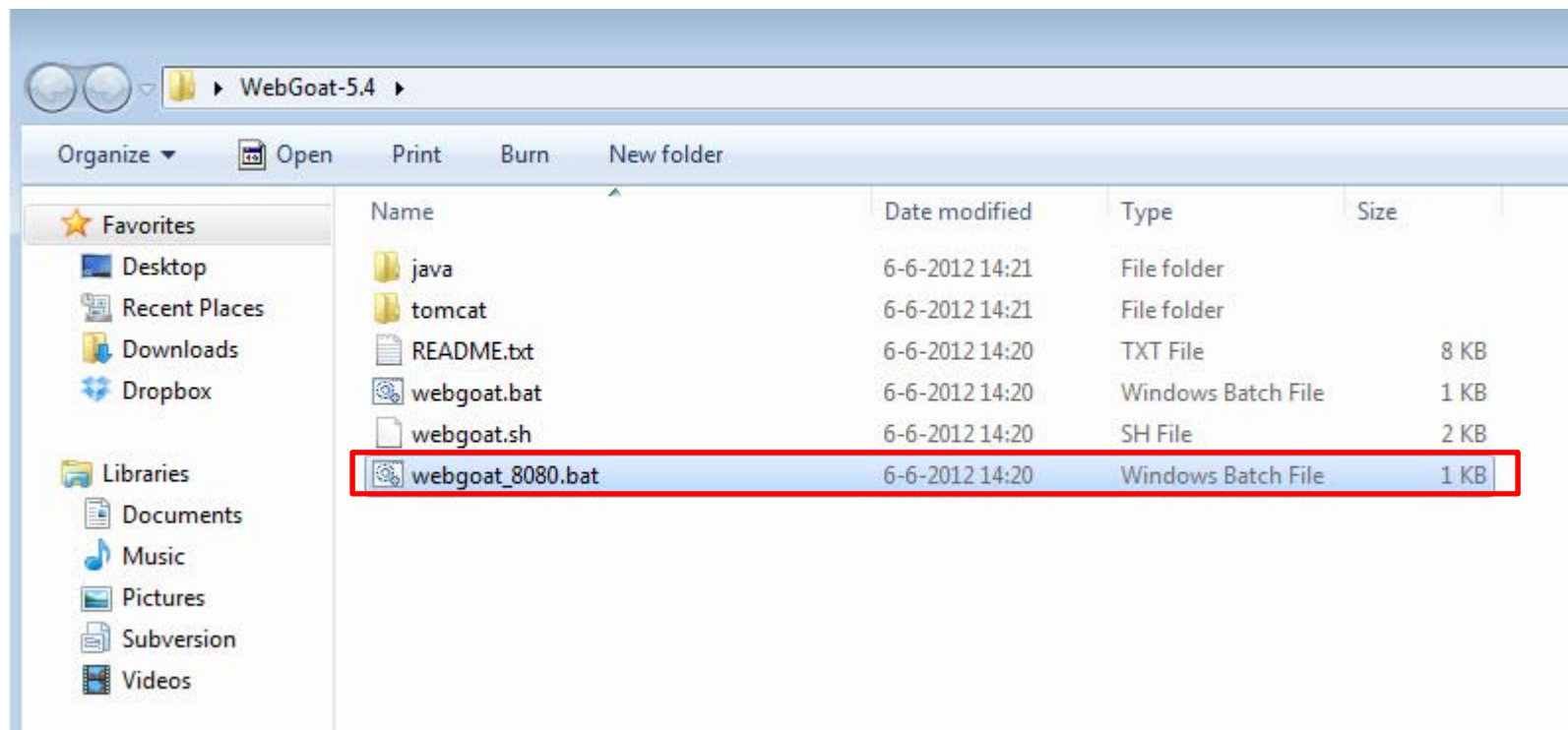
Tamper Data

- 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



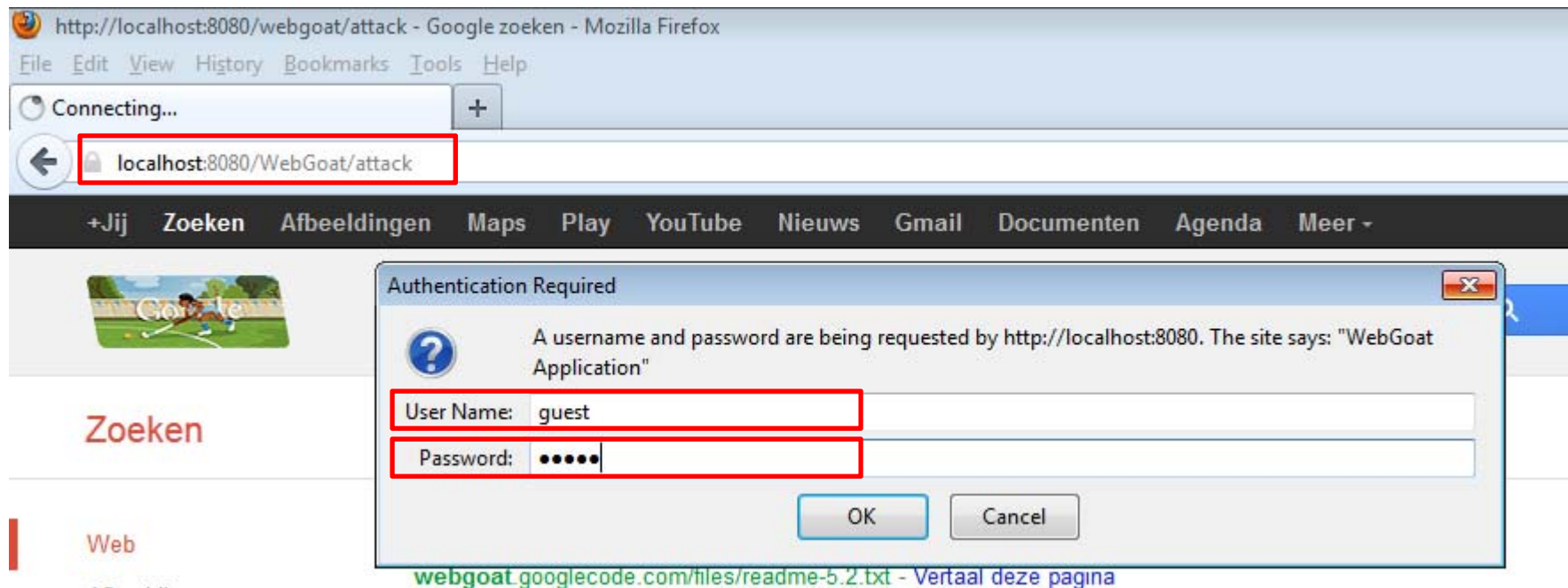
WEB GOAT (1)

- 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*

The image displays two screenshots of the OWASP WebGoat v5.4 web application interface.

Left Screenshot: Main Landing Page

- Browser: Mozilla Firefox, URL: localhost:8080/WebGoat/attack
- Header: OWASP WebGoat v5.4
- Text: "Thank you for using WebGoat! This program is a demonstration of common web application security exercises intended to provide hands on experience with application penetration testing." "The WebGoat project is led by Bruce Mayhew. Please send all comments to Bruce Mayhew at WebGoat@owasp.org."
- Logos: OWASP (The Open Web Application Security Project) and ASPECT SECURITY (Application Security).
- WebGoat Authors: Bruce Mayhew, Jeff Williams
- WebGoat Design Team: David Anderson, Laurence Casey (Graphics), Rogan Dawes, Bruce Mayhew
- V5.4 Lesson Contributors: Sherif Koussa, Yiannis Pavlosoglou
- Documentation Contributors: Erwin Geirnaert, Aung Khant, Sherif Koussa
- Special Thanks for V5.4: Brian Ciomei (Multitude of bug fixes) To all who have sent comments
- Buttons: **Start WebGoat** (highlighted with a red box)
- Warning: "WARNING While running this program, your machine is extremely vulnerable to attack if you are not running it on localhost (default configuration). You should disconnect from the network while using this program. This program is for educational purposes only. Use of these techniques without permission could result in financial liability, and/or criminal penalties."

Right Screenshot: How to work with WebGoat

- Browser: Mozilla Firefox, URL: localhost:8080/WebGoat/attack
- Header: OWASP WebGoat v5.4
- Language: Choose another language: English
- Buttons: Logout, Hints, Show Params, Show Cookies, Lesson Plan, Show Java, Solution
- Section: **How to work with WebGoat**
- Text: "Welcome to a short introduction to WebGoat. Here you will learn how to use WebGoat and additional tools for the lessons." "Environment Information WebGoat uses the Apache Tomcat server. It is configured to run on localhost although this can be easily changed. This configuration is for single user, additional users can be added in the tomcat-users.xml file. If you want to use WebGoat in a laboratory or in class you might need to change this setup. Please refer to the Tomcat Configuration in the Introduction section." "The WebGoat Interface"
- Table of Contents (highlighted with a red box):
 - Introduction
 - General
 - Access Control Flaws
 - AJAX Security
 - Authentication Flaws
 - Buffer Overflows
 - Code Quality
 - Concurrency
 - Cross-Site Scripting (XSS)
 - Improper Error Handling
 - Injection Flaws
 - Denial of Service
 - Insecure Communication
 - Insecure Configuration
 - Insecure Storage
 - Malicious Execution
 - Parameter Tampering
 - Session Management Flaws
 - Web Services
 - Admin Functions
 - Challenge
- Buttons: Restart this Lesson
- Footer: OWASP WebGoat V5.2, Hints, Show Params, Show Cookies, Lesson Plan, Show Java, Solution

LAB SESSION 2

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

Choose another language: English ▼ Logout ?

OWASP WebGoat v5.4 ◀ Hints ▶ Show Params Show Cookies Lesson Plan Show Java Solution

Http Basics

Introduction
General
[Http Basics](#)
[HTTP Splitting](#)
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service

Solution Videos Restart this Lesson

Enter your name in the input field below and press "go" to submit. The server will accept the request, reverse the input, and display it back to the user, illustrating the basics of handling an HTTP request.

The user should become familiar with the features of WebGoat by manipulating the above buttons to view hints, show the HTTP request parameters, the HTTP request cookies, and the Java source code. You may also try using WebScarab for the first time.

Enter your Name:

- *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.

The screenshot shows the OWASP WebGoat v5.4 web application interface on the left and a Tamper Popup dialog box on the right. The web application has a navigation menu on the left with categories like 'Introduction', 'General', and 'Http Basics'. The main content area displays 'Solution Videos' with instructions: 'Enter your name in the input field below and press the Go! button to view hints, reverse the input, and display it back to the user via an HTTP request.' Below this is an input field containing 'elisa' and a 'Go!' button. The Tamper Popup dialog box is titled 'http://localhost:8080/WebGoat/attack?Screen=16&menu=100' and contains two tables. The first table lists request headers, and the second table lists post parameters. The 'person' parameter is highlighted in blue, and its value 'elisa' is also highlighted in blue. The 'SUBMIT' parameter has a value of 'Go%21'. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

OWASP WebGoat v5.4

Choose another language: English

Introduction
General
[Http Basics](#)
[HTTP Splitting](#)
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Session Management Flaws
Web Services
Admin Functions
Challenge

[Hints](#) [Show Params](#) [Show Cookies](#) [Le...](#)

Solution Videos

Enter your name in the input field below and press the Go! button to view hints, reverse the input, and display it back to the user via an HTTP request.

The user should become familiar with the features of the application by clicking the buttons to view hints, show the HTTP request parameters, and view the Java source code. You may also try using WebScarab.

Enter your Name:

OWASP Foundation | Project WebGoat | Report Bug

2/22/2013

Tamper Popup

http://localhost:8080/WebGoat/attack?Screen=16&menu=100

Request Header Name	Request Header V...	Post Parameter Name	Post Parameter V...
Host	localhost:8080	person	<input type="text" value="elisa"/>
User-Agent	Mozilla/5.0 (Windc	SUBMIT	<input type="text" value="Go%21"/>
Accept	text/html,applicati		
Accept-Language	en-us,en;q=0.5		
Accept-Encoding	gzip, deflate		
Connection	keep-alive		
Referer	http://localhost:80		
Cookie	JSESSIONID=912C:		
Authorization	Basic Z3Vlc3Q6Z3V		

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

The image shows a screenshot of the OWASP WebGoat v5.4 application. On the left, a list of vulnerabilities is displayed, with 'Insecure Login' highlighted in blue. A red box highlights 'Insecure Communication' in the list, and a red line points from this box to the 'Insecure Login' link in the list. The main content area shows the 'Insecure Login' lesson page, which includes a 'Please Login' form with fields for 'Enter your name:' (containing 'Jack') and 'Enter your password:' (containing seven dots), and a 'Submit' button. The page title is 'Goat Hills Financial Human Resources'.

- **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”

The screenshot displays the OWASP WebGoat v5.4 interface. On the left, a navigation menu lists various security topics, with "Insecure Login" highlighted in a red box. The main content area shows a "Goat Hills Financial Human Resources" login form with fields for "Enter your name" (containing "Jack") and "Enter your password" (containing "*****"), and a "Submit" button. A "Tamper Popup" window is overlaid on the right, showing the URL "http://localhost:8080/WebGoat/attack?Screen=67&menu=1300". The popup contains two tables: "Request Header Name" and "Request Header V...", and "Post Parameter Name" and "Post Parameter V...". The "Post Parameter Name" table has a red box around the "clear_user" row, which has the value "Jack". The "clear_pass" row has the value "sniffy". The "Submit" row has the value "Submit". The date "2/22/2013" is visible at the bottom of the page.

Request Header Name	Request Header V...	Post Parameter Name	Post Parameter V...
Host	localhost:8080	clear_user	Jack
User-Agent	Mozilla/5.0 (Windc	clear_pass	sniffy
Accept	text/html,applicati	Submit	Submit
Accept-Language	en-us,en;q=0.5		
Accept-Encoding	gzip, deflate		
Connection	keep-alive		
Referer	http://localhost:80		
Cookie	JSESSIONID=912C:		
Authorization	Basic Z3Vlc3Q6Z3Vl		

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

Choose another language: English Logout ?

Exploit Hidden Fields

OWASP WebGoat v5.4 < Hints > Show Params Show Cookies Lesson Plan Show Java Solution

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Bypass HTML Field Restrictions
Exploit Hidden Fields
Exploit Unchecked Email
Bypass Client Side JavaScript Validation
Session Management Flaws
Web Services
Admin Functions
Challenge

Solution Videos Restart this Lesson

Try to purchase the HDTV for less than the purchase price, if you have not done so already.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
56 inch HDTV (model KTV-551)	2999.99	1	\$2999,99

The total charged to your credit card: \$2999,99

ASPECT SECURITY
Application Security Experts

OWASP Foundation | Project WebGoat | Report Bug

- *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

The screenshot displays the OWASP WebGoat v5.4 interface. The main content area shows a 'Shopping Cart' with one item: '56 inch HDTV (model KTV-551)' priced at \$2999.99 with a quantity of 1, totaling \$2999.99. Below the cart, there are 'UpdateCart' and 'Purchase' buttons. A 'Tamper Popup' window is overlaid on the right, showing the current request headers and post parameters. The URL is 'http://localhost:8080/WebGoat/attack?Screen=34&menu=1700'. The 'Post Parameter Name' column shows 'Price' with a value of '1.00', indicating the tampering. The 'Request Header Name' column shows various headers like Host, User-Agent, Accept, etc.

OWASP WebGoat v5.4 | Exploit Hidden Fields

Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering

Solution Videos | Restart this Lesson

Try to purchase the HDTV for less than the purchase price, if you have not done so already.

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
56 inch HDTV (model KTV-551)	2999.99	1	\$2999.99

The total charged to your credit card: \$2999.99 [UpdateCart] [Purchase]

ASPECT SECURITY
Application Security Experts

OWASP Foundation | Project WebGoat | Report Bug

Tamper Popup
http://localhost:8080/WebGoat/attack?Screen=34&menu=1700

Request Header Name	Request Header V...	Post Parameter Name	Post Parameter V...
Host	localhost:8080	QTY	1
User-Agent	Mozilla/5.0 (Windc	SUBMIT	Purchase
Accept	text/html,applicati	Price	1.00
Accept-Language	en-us,en;q=0.5		
Accept-Encoding	gzip, deflate		
Connection	keep-alive		
Referer	http://localhost:80		
Cookie	JSESSIONID=912C:		
Authorization	Basic Z3Vlc3Q6Z3V		

OK Cancel

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