

# Seguretat Informàtica (SI)

## LAB 1 - Análisis de vulnerabilidades web

Francisco Jordan <jordan@ac.upc.edu>

QP20 – Abril 2020

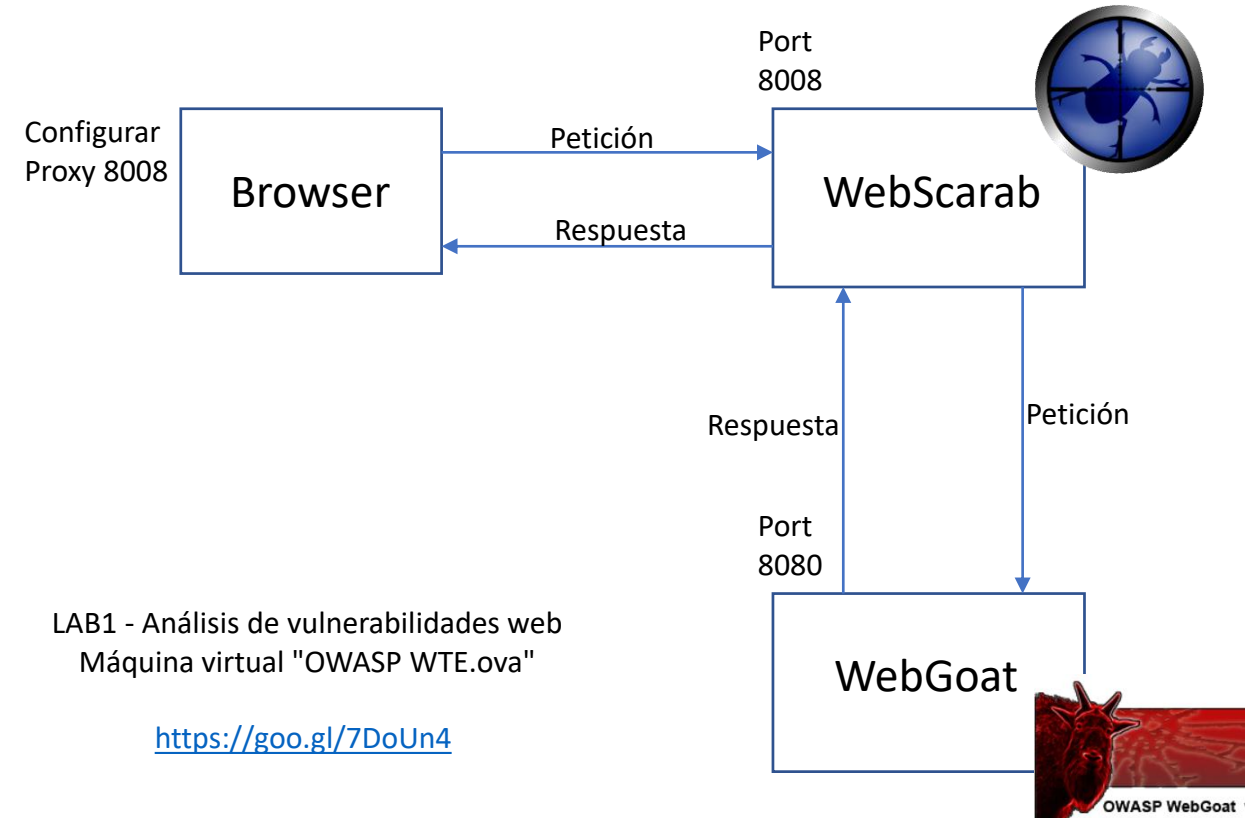
# Objetivos del laboratorio

- Analizar diversas vulnerabilidades sobre un entorno de aplicación real
- Aprender a utilizar herramientas de opensource del proyecto OWASP (The Open Web Application Security Project) para practicar sobre vulnerabilidades conocidas en aplicaciones reales
- Poner en práctica los conocimientos de auditoría de aplicaciones web adquiridos en clase

# Esquema de la práctica

**WebGoat** is a J2EE application developed by OWASP (The Open Web Application Security Project) and based on Tomcat. It is an insecure application and it is basically its purpose. The objective is to use it as an introduction to different attacks directed to web applications (test environment). It has different lessons that provide us with help and information to understand and to be able to overcome them.

**WebScarab** is a framework to analyze web applications developed by OWASP. It uses http and https and it can be used as a proxy to study a web page requests and responses, review and modify them before they get to the client or the server.



# Software a utilizar y notas introducción

- Descargar la imagen de la máquina virtual “OWASP WTE.ova” que hay en <https://goo.gl/7DoUn4> e importarla en Vmware. Con 2GB de RAM y 2 CPUs va sobrada. El usuario y password son ambos “si2018”
- Seguir las instrucciones en el documento guía “[GUÍA] LAB 1 - Analisis de vulnerabilidades web.pdf”. Fijarse en las notas incluidas en la guía
- Una vez arrancado el WebGoat, leer la sección de introducción dentro de la aplicación “How to work with WebGoat” (ver captura slide siguiente)
- Leer el punto de “Entrega” de esta presentación y antes de empezar los ejercicios prácticos leer las preguntas del formulario a entregar
- A continuación tenéis algunas notas prácticas para desarrollar los ejercicios



OWASP WebGoat V5.3

◀ Hints ▶ Show Params Show Cookies Lesson Plan Show Java Solution

# How to work with WebGoat

Introduction

Solution Videos

Restart this Lesson



- ✓ [How to work with WebGoat](#)
- [Tomcat Configuration](#)
- [Useful Tools](#)
- [Create a WebGoat Lesson](#)

## How To Work With WebGoat

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

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



# Ejercicios a realizar

- Los ejercicios propuestos siguen la nomenclatura de la aplicación WebGoat tanto en la categoría como en el nombre del ejercicio (ver slide siguiente). Al lado del ejercicio se incluye el título del punto de la guía “[GUÍA] LAB 1 - Analisis de vulnerabilidades web.pdf” que detalla dicho ejercicio.
- **NOTA.** *La versión de la aplicación que hay en la guía es más antigua así que pueden diferir un poco las páginas, pero las instrucciones siguen siendo válidas*
- Categoría: Parameter Tampering
  - Exploit Hidden Fields -> 1.3.1.1 Hidden fields
  - Exploit Unchecked Email -> 1.3.1.2 e-mail not validated
  - Bypass Client Side JavaScript Validation -> 1.3.1.3 Avoid validations on the client side
- Categoría: Improper Error Handling
  - Fail Open Authentication Scheme -> 1.3.1.4 Fail Open Authentication Scheme
- Categoría: Session Management Flaws
  - Spoof an Authentication Cookie -> 1.3.2.1 Authentication using cookies
- Categoría: Injection Flaws
  - String SQL Injection -> 1.3.4 SQL Injection
- Categoría: AJAX Security
  - JSON Injection -> NO tiene referencia en la guía



Exploit Hidden Fields - Mozilla Firefox

File Edit View History Bookmarks Tools Help

← → ↺ × 🏠 🖼️


http://127.0.0.1:8080/webgoat/attack?Screen=60&menu=1700

★ Google 🔍

Most Visited ▾ Getting Started 📡 Latest Headlines ▾

Exploit Hidden Fields +

Choose another language: English ▾ Logout ?



OWASP WebGoat V5.3

Exploit Hidden Fields

⏪ 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)  
Denial of Service  
Improper Error Handling  
Injection Flaws  
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	<input type="text" value="1"/>	\$2999.99

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

ASPECTSECURITY  
Application Security Specialists

OWASP Foundation | Project WebGoat | Report Bug

# NOTAS Práctica

- En el ejercicio “AJAX Security/JSON Injection” se utiliza el objeto javascript XMLHttpRequest del navegador para invocar al servidor en una URL (página) concreta de forma transparente a la página actual visible en el navegador. OJO porque este objeto se puede invocar múltiples veces prevaleciendo el último resultado



# Entrega

- La entrega de esta práctica consta de 2 partes:
  - Completar un formulario on-line con 10 preguntas publicado en Atenea junto a los documentos de Presentación y Guía, y
  - Subir un fichero “si-q20-lab1.tar” que contenga los ficheros con nombres “formulario.txt”, “p1.png”, “p3.gif”, ... “p10.jpg” (total 10 ficheros ya que la pregunta 2 no tiene captura)
  - Los ficheros captura pueden ser cualquier formato gráfico estándar, donde cada uno contiene la captura de pantalla de la máquina virtual con la siguiente información visible:
    - Captura completa venta VM incluyendo el nombre de la imagen virtual
    - Datos del alumno (tal y como aparecen en el ejemplo siguiente)
    - Fecha actual de la captura que marca el sistema (hora local real)
    - El navegador/aplicación WebGoat con la información que se pide de “Respuesta” en el cuestionario visible
- **NOTA.** En la slide siguiente tenéis un ejemplo de captura en el que, por ejemplo, en el cuestionario se pide introducir las ciudades (son un ejemplo ilustrativo que no obedece al caso real del ejercicio) from y to que aparecen en el ejercicio “JSON Injection”
- El fichero “formulario.txt” contiene líneas diferentes para cada respuesta que se incluirán en este fichero copiada igual que en el formulario. Por ejemplo, “**Pregunta1 – Respuesta: texto respuesta**” y así tantas líneas como Pregunta/Respuesta haya en el formulario

SI - QP20  
GRUPO-1X  
FRANCISCO JORDAN  
DNI-11222333



OWASP WebGoat V5.3

◀ Hints ▶ Show Params Show Cookies Lesson Plan Show Java Solution

Introduction  
General  
Access Control Flaws  
AJAX Security

[LAB: DOM-Based cross-site scripting](#)  
[LAB: Client Side Filtering](#)  
[Same Origin Policy Protection](#)  
[DOM Injection](#)  
[XML Injection](#)  
[JSON Injection](#)  
[Silent Transactions Attacks](#)  
[Dangerous Use of Eval](#)  
[Insecure Client Storage](#)

Authentication Flaws  
Buffer Overflows  
Code Quality  
Concurrency  
Cross-Site Scripting (XSS)  
Denial of Service  
Improper Error Handling  
Injection Flaws  
Insecure Communication  
Insecure Configuration  
Insecure Storage  
Malicious Execution  
Parameter Tampering  
Session Management Flaws

### Solution Videos

Restart this Lesson

- \* You are traveling from Boston, MA- Airport code BOS to Seattle, WA - Airport code SEA.
- \* Once you enter the three digit code of the airport, an AJAX request will be executed asking for the ticket price.
- \* You will notice that there are two flights available, an expensive one with no stops and another cheaper one with 2 stops.
- \* Your goal is to try to get the one with no stops but for a cheaper price.

From:

Barcelona

To:

Los Angeles

Submit

Created by Sherif Koussa 

OWASP Foundation | Project WebGoat | Report Bug

# Formulario on-line

- Se debe responder a las preguntas de forma exacta (case-sensitive) tal y como se pide. El formulario se corrige automáticamente
- La evaluación del formulario sigue las siguientes pautas:
  - Cada pregunta debe responderse obligatoriamente y después enviar el formulario entero para su valoración
  - Se permiten un número ilimitado de intentos por pregunta, pero cada intento incorrecto penaliza 1/3 de la puntuación
  - La valoración de cada pregunta es de 1 punto de partida. La valoración final es la media simple de todos los intentos
  - La valoración final del formulario es la suma de la valoración de las preguntas, siendo 5 puntos la nota de aprobado
- Las repuestas del formulario se comprobarán con las evidencias y respuestas del fichero de entrega “si-q20-lab1.tar”

# Fechas importantes y seguimiento

- Publicación de la práctica: 27/04/2020
- Fecha límite de entrega: 18/5/2020
- Se abre un foro en el racó para la práctica “LAB 1 - Análisis de vulnerabilidades web”. Cualquier duda o pregunta publicarlo en el foro o enviarme un email. Acordaremos una sesión on-line de dudas