

PROGRAMACIÓN ORIENTADA A PRUEBAS

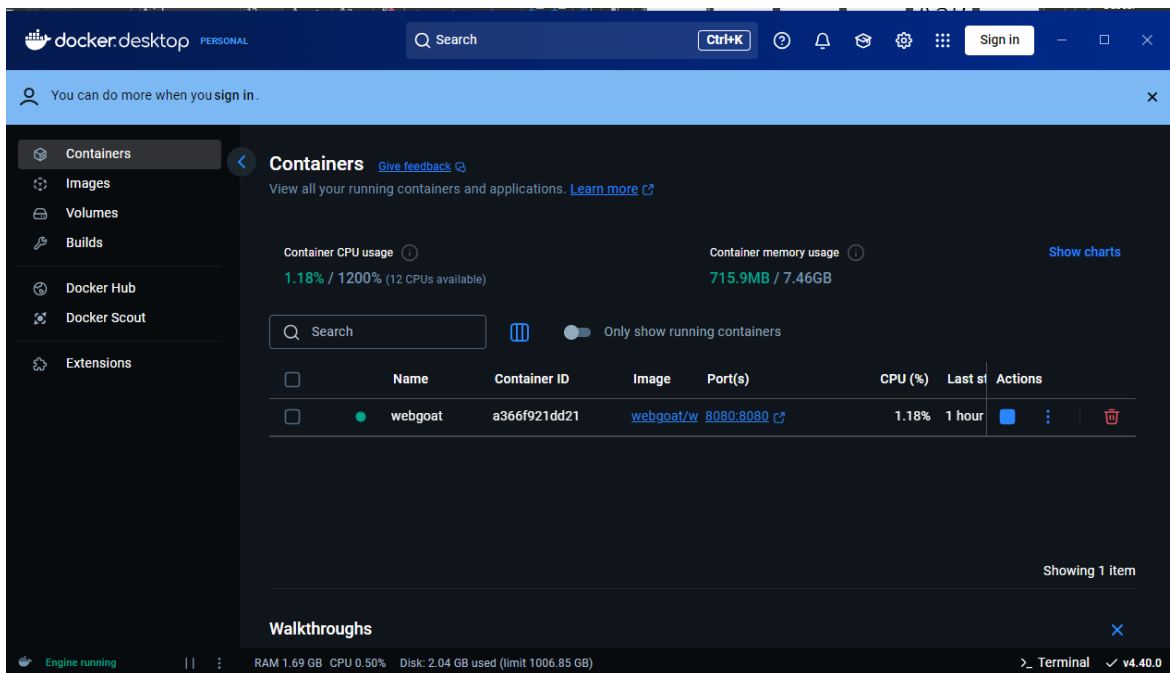
Revisado por: Ing. Miguel Cardona.

Aarón Hernández García.

Instalación y Uso de OWASP WebGoat con Docker

09/04/2025

Instalar Docker en tu máquina o utilizar un entorno con Docker preinstalado.



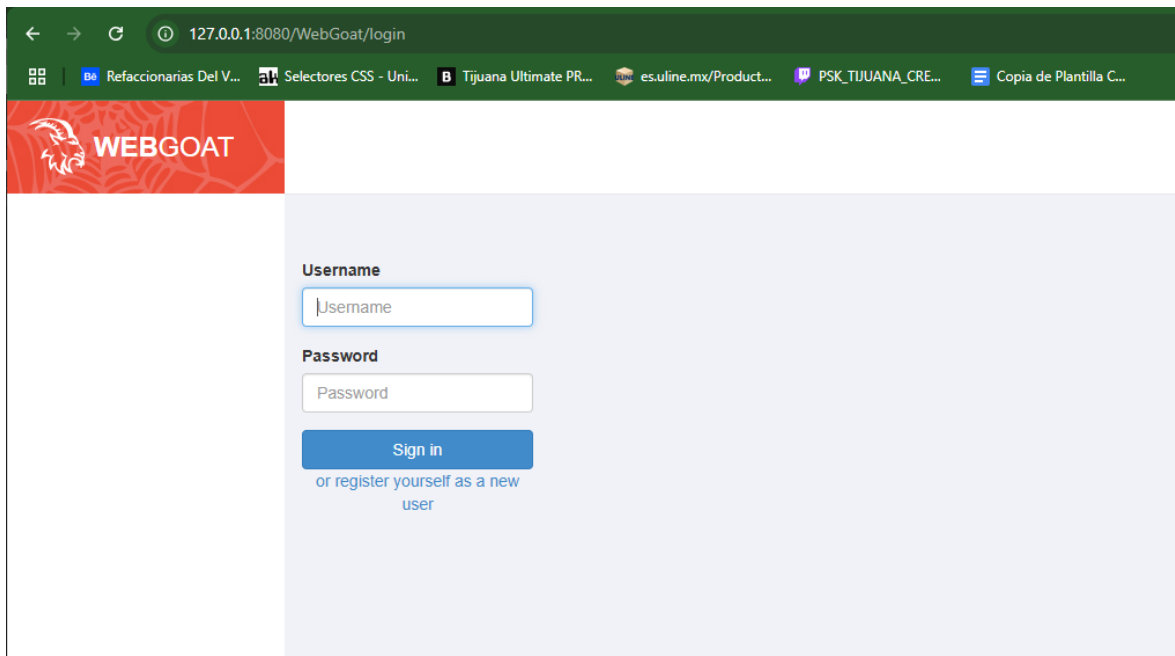
Ejecutar el siguiente comando para descargar y ejecutar WebGoat:docker

run -d -p 8080:8080 --name webgoat webgoat/webgoat

```
C:\Windows\system32>docker run -d -p 8080:8080 --name webgoat webgoat/webgoat
Unable to find image 'webgoat/webgoat:latest' locally
latest: Pulling from webgoat/webgoat
4f4fb700ef54: Pull complete
7b2e4df376c9: Pull complete
7b5a3507f742: Pull complete
307ac9d4e999: Pull complete
d1504bee8985: Pull complete
ea28f6f7f0aa: Pull complete
5a7813e071bf: Pull complete
Digest: sha256:3101bd9e7bcfe122d7ef91e690ef3720de36cc4e86b3d06763a1ddf2e2751a4b
Status: Downloaded newer image for webgoat/webgoat:latest
a366f921dd21c61ae2a153585b4f85a50146c36f9f5a4e0e2eb1d39143635494
C:\Windows\system32>
```

Acceder a WebGoat en tu navegador ingresando la URL:

http://localhost:8080/WebGoat



← → ↻ 127.0.0.1:8080/WebGoat/login

Refaccionarias Del V... Selectores CSS - Uni... Tijuana Ultimate PR... es.uline.mx/Product... PSK_TIJUANA_CRE... Copia de Plantilla C...

WEBGOAT

Username

Password

Sign in

[or register yourself as a new user](#)

Register

Username

Password

Confirm password

Terms of use


While running this program your machine will be extremely vulnerable to attack. You should disconnect from the Internet while using this program. WebGoat's default configuration binds to localhost to minimize the exposure.

This program is for educational purposes only. If you attempt these techniques without authorization, you are very likely to get caught. If you are caught engaging in unauthorized hacking, most companies will fire you. Claiming that you were doing security research will not work as that is the first thing that all hackers claim.

☒ Agree with the terms and conditions


Sign up

Explorar las lecciones disponibles, enfocándose en vulnerabilidades como: SQL Injection, Cross-Site Scripting (XSS), Authentication Bypass.

**WEBGOAT**

WebGoat

Search lesson



Introduction >

General >

(A1) Broken Access Control >

(A2) Cryptographic Failures >

(A3) Injection >

(A5) Security Misconfiguration >

(A6) Vuln & Outdated Components >

(A7) Identity & Auth Failure >

(A8) Software & Data Integrity >

(A9) Security Logging Failures >

(A10) Server-side Request Forgery >

Client side >

Challenges >

Reset lesson

1

What is WebGoat?

Welcome **sogecool** !

WebGoat is a deliberately insecure application that allows interested developers just like you to *test vulnerabilities* commonly found in Java-based applications that use common and popular open source components.

Now, while we in no way condone causing intentional harm to any animal, goat or otherwise, we think learning everything you can about security vulnerabilities is essential to understanding just what happens when even a small bit of unintended code gets into your applications.

What better way to do that than with your very own scapegoat?

Feel free to do what you will with him. Hack, poke, prod and if it makes you feel better, scare him until your heart's content. Go ahead, and hack the goat. We promise he likes it.

Thanks for your interest!

The WebGoat Team

Cross-Site Request Forgery (CSRF)

1. Introducción

Cross-site request forgery, abreviado CSRF (o a veces XSRF) y conocido también como one-click attack o session riding, es un tipo de vulnerabilidad web en la que se aprovecha la confianza que un sitio web deposita en el navegador de un usuario autenticado. En lugar de explotar la confianza que el usuario tiene en un sitio (como ocurre en el cross-site scripting o XSS), CSRF se aprovecha de la confianza que el sitio tiene en el navegador del usuario, induciendo a este último a enviar peticiones HTTP no autorizadas a un sitio en el que ya se ha autenticado.

2. Descripción y Mecanismo de CSRF

2.1. Definición y Concepto

Una falsificación de petición en sitios cruzados es un ataque de "deputado confundido" (confused deputy attack) que se dirige a aplicaciones web. El atacante engaña al navegador del usuario para que envíe solicitudes HTTP a un sitio de confianza sin que el usuario tenga conocimiento de ello. Estas solicitudes pueden incluir acciones que modifican el estado en el servidor, como cambiar una dirección de correo electrónico, contraseña o realizar una transacción.

2.2. Características Comunes

Las principales características de un ataque CSRF son:

- **Dependencia de la Identidad del Usuario:**
El ataque se dirige a sitios que confían en la identidad del usuario, normalmente porque el usuario ha iniciado sesión y su autenticación se mantiene mediante cookies o tokens de sesión.
- **Explotación de la Confianza del Sitio:**
Se aprovecha de la confianza que el sitio confiere a las solicitudes que provienen del navegador autenticado. El atacante no necesita robar credenciales; en cambio, induce al navegador a actuar en nombre del usuario.
- **Engaño del Navegador:**
El atacante manipula la forma en que se envían las solicitudes HTTP desde el navegador del usuario, aprovechándose de scripts maliciosos o enlaces diseñados para desencadenar acciones indeseadas.
- **Solicitudes HTTP con Efectos Laterales:**
El objetivo del CSRF son las solicitudes que cambian el estado en el

servidor. Esto puede incluir modificaciones en datos sensibles, transferencia de fondos, cambios en configuraciones o cualquier otra acción con efectos colaterales en el sistema.

3. Comparación con Otras Vulnerabilidades

3.1. CSRF vs. XSS

- **Cross-Site Scripting (XSS):**
En un ataque XSS, el atacante explota la confianza del usuario en el sitio web al inyectar código malicioso que se ejecuta en el navegador del usuario. Esto puede resultar en el robo de información, manipulación del contenido o redirección de la sesión.
- **Cross-Site Request Forgery (CSRF):**
En contraste, CSRF explota la confianza que el sitio web tiene en el navegador del usuario. El usuario ya autenticado realiza solicitudes en segundo plano que el atacante no ve ni controla la respuesta; sin embargo, estas solicitudes pueden cambiar el estado en el servidor.

Esta diferencia es crucial para comprender qué controles y validaciones deben implementarse para prevenir cada tipo de ataque.

4. Escenarios de Riesgo y Ejemplos de Impacto

4.1. Ejemplo de Acción No Autorizada

Un escenario típico de CSRF es cuando un usuario inicia sesión en un sitio bancario. El sitio utiliza cookies para mantener la sesión. Si el usuario visita, sin saberlo, una página maliciosa controlada por un atacante, ésta podría enviar una solicitud HTTP al banco para transferir fondos a una cuenta que el atacante controla. Como el navegador incluye las cookies de sesión, el banco procesa la solicitud como si fuera legítima.

4.2. Acciones con Efectos Laterales

- **Cambio de información personal:**
El atacante podría modificar la dirección de correo electrónico o la contraseña de un usuario.
- **Transacciones no autorizadas:**
En aplicaciones financieras, se podrían realizar transferencias de fondos.
- **Modificación de configuraciones:**
Aplicaciones que permiten cambiar configuraciones sensibles podrían ser afectadas sin el consentimiento explícito del usuario.

5. Medidas de Mitigación y Prevención

Para proteger una aplicación contra ataques CSRF se recomiendan las siguientes prácticas:

5.1. Tokens CSRF (Anti-CSRF Tokens)

- **Generación y Validación:**
Se deben generar tokens únicos para cada sesión de usuario que se incluyan en los formularios o en las peticiones sensibles. El servidor debe validar que el token recibido en la petición coincida con el token almacenado en la sesión.
- **Implementación en Formularios:**
Incluir un campo oculto que contenga el token y, en el servidor, comprobarlo antes de procesar la solicitud.

5.2. Uso de Cabeceras Personalizadas

- **Verificación de origen:**
Configurar cabeceras HTTP personalizadas (por ejemplo, X-Requested-With) para diferenciar solicitudes legítimas desde la propia aplicación de aquellas que provienen de otros orígenes.

5.3. Políticas SameSite en Cookies

- **Configuración de las cookies:**
Establecer el atributo SameSite en las cookies de sesión para restringir su envío en peticiones de origen cruzado. Esto previene que el navegador envíe cookies en solicitudes iniciadas desde otros sitios.

5.4. Re-validación de Acciones Sensibles

- **Solicitar confirmación:**
Para acciones críticas (como cambios de contraseña o transferencias financieras), se puede requerir que el usuario vuelva a autenticarse o confirme explícitamente la acción, reduciendo el riesgo de acciones automatizadas.

6. Ejercicios Prácticos para Comprender CSRF

Se recomienda realizar algunos ejercicios prácticos para entender mejor cómo opera un ataque CSRF y cómo prevenirlo. Estos pueden incluir:

- **Simulación de una solicitud CSRF:**
Crear una pequeña página web que, al visitarla, envíe una petición POST a

una aplicación vulnerable que utilice cookies para mantener la autenticación.

- **Implementación de tokens Anti-CSRF:**
Desarrollar un formulario web en el que se genere y valide un token CSRF de manera correcta, y comprobar su eficacia mediante pruebas.
- **Configuración de cookies con SameSite:**
Modificar la configuración de las cookies de sesión de una aplicación para incluir el atributo SameSite y evaluar el comportamiento en diferentes navegadores.

7. Conclusión

El Cross-Site Request Forgery (CSRF) es un ataque sofisticado que aprovecha la confianza que un sitio web deposita en el navegador del usuario. Al forzar al usuario a enviar solicitudes HTTP maliciosas mediante técnicas de engaño, un atacante puede provocar acciones no autorizadas en aplicaciones confiables. Entender cómo opera un CSRF es fundamental para implementar defensas apropiadas, como el uso de tokens CSRF, la configuración adecuada de las cookies y la validación de cabeceras personalizadas.

Adoptar estas medidas no solo protege la aplicación contra ataques CSRF, sino que también mejora la robustez general del sistema frente a otras amenazas de seguridad.