
Auditoría WebGoat (KeepCoding)

David Groning Hernández



KEEPCODING

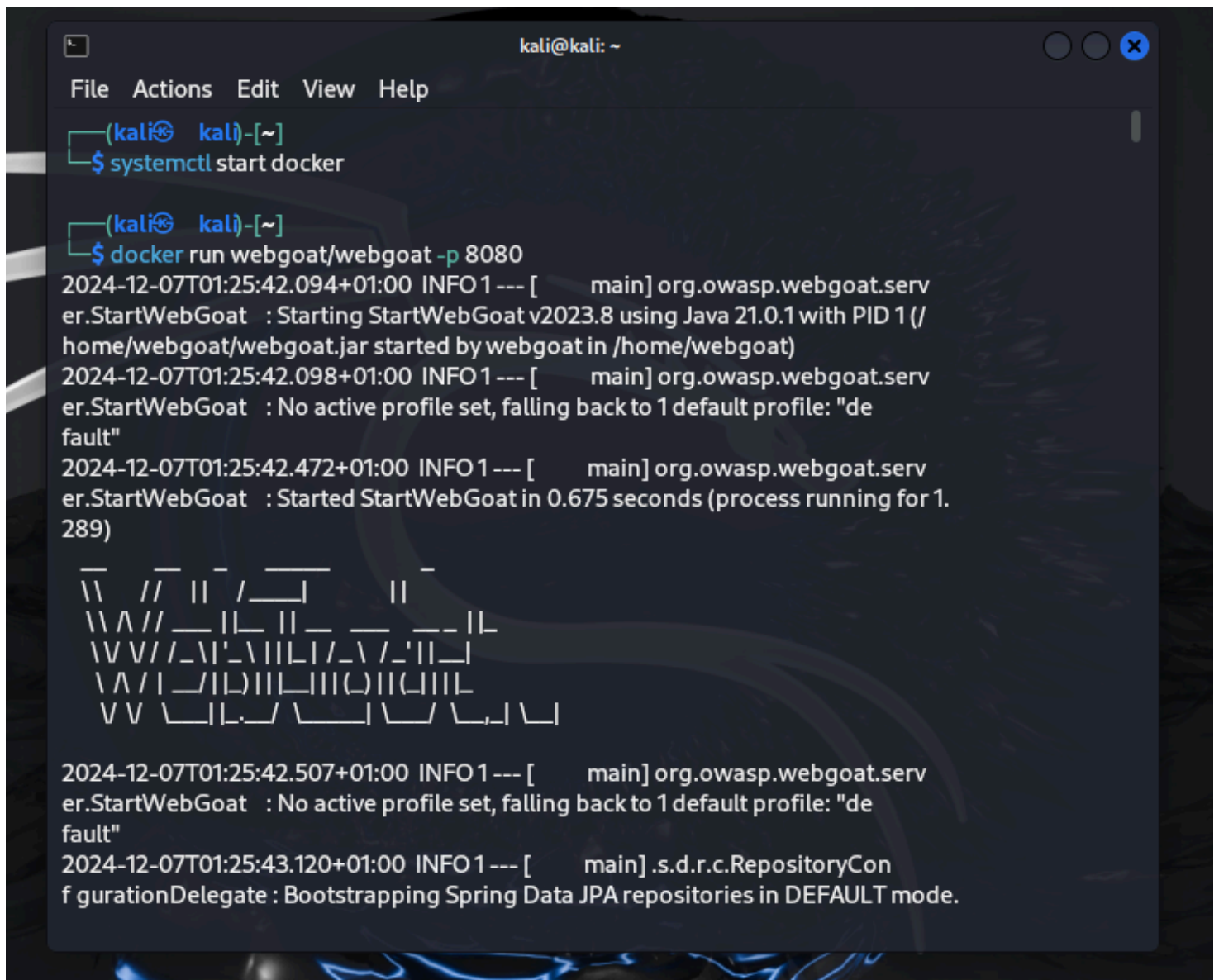
Tech School



WEBGOAT

Inicio de la maquina virtual y docker

- `systemctl start docker.`
- `docker run webgoat/webgoat -p 8080`
(-p 8080 para iniciarlo en el dicho puerto).



```
kali@kali: ~  
File Actions Edit View Help  
(kali) [~]  
$ systemctl start docker  
  
(kali) [~]  
$ docker run webgoat/webgoat -p 8080  
2024-12-07T01:25:42.094+01:00 INFO 1 --- [      main] org.owasp.webgoat.server.StartWebGoat : Starting StartWebGoat v2023.8 using Java 21.0.1 with PID 1 (/home/webgoat/webgoat.jar started by webgoat in /home/webgoat)  
2024-12-07T01:25:42.098+01:00 INFO 1 --- [      main] org.owasp.webgoat.server.StartWebGoat : No active profile set, falling back to 1 default profile: "default"  
2024-12-07T01:25:42.472+01:00 INFO 1 --- [      main] org.owasp.webgoat.server.StartWebGoat : Started StartWebGoat in 0.675 seconds (process running for 1.289)  
  
  _ _ _ _ _  
 \_//  || /__|  ||  
  \_//  ||  ||  ||  
  \_//  ||  ||  ||  
  \_//  ||  ||  ||  
  \_//  ||  ||  ||  
  \_//  ||  ||  ||  
  \_//  ||  ||  ||  
  \_//  ||  ||  ||  
  \_//  ||  ||  ||  
  
2024-12-07T01:25:42.507+01:00 INFO 1 --- [      main] org.owasp.webgoat.server.StartWebGoat : No active profile set, falling back to 1 default profile: "default"  
2024-12-07T01:25:43.120+01:00 INFO 1 --- [      main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
```

A3 - Injection

- SQL Injection

- -Para identificar si una página es vulnerable o buscar indicios de esto se prueba con una comilla “ ‘ ” y depende de lo que nos devuelva podemos determinar el nivel de seguridad que presentan.
- -Al ser vulnerable podemos empezar a probar con una inyección SQL básica “ ‘ or 1=1 “.

It is your turn!

You are an employee named John **Smith** working for a big company. The company has an internal system that allows all employees to see their own internal data such as the department they work in and their salary.

The system requires the employees to use a unique *authentication TAN* to view their data.
Your current TAN is **3SL99A**.

Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, *you want to take a look at the data of all your colleagues* to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need.

You already found out that the query performing your request looks like this:

```
"SELECT * FROM employees WHERE last_name = '' + name + '' AND auth_tan = '' + auth_tan + ''";
```

Employee Name:

Authentication TAN:

Sorry the solution is not correct, please try again.

malformed string: ' AND auth_tan = "

- Esta presenta un error y se le añade “ -- “ al final del código ya que esta hace que emita el código que sigue corriendo en el backend.

Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, you want to take a look at the data of all your colleagues to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need.

You already found out that the query performing your request looks like this:

```
"SELECT * FROM employees WHERE last_name = ' ' + name + ' ' AND auth_tan = ' ' + auth_tan + ' '";
```



Employee Name:

Authentication TAN:

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN
32147	Paulina	Travers	Accounting	46000	P45JSI
34477	Abraham	Holman	Development	50000	UU2ALK
37648	John	Smith	Marketing	64350	3SL99A
89762	Tobi	Barnett	Development	77000	TA9LL1
96134	Bob	Franco	Marketing	83700	LO9S2V

- Una vez aplicado podemos comprobar que la inyección se ha realizado con éxito, dándonos la oportunidad de conseguir la información que buscábamos o podríamos realizar más operaciones.

A3 – Injection

- **Cross Site Scripting**

- Nos presentan una situación donde hay dos interfaces y nos sugieren usar un ataque XSS con “alert()” o “console.log()”.
- Tras saber lo anterior aplicamos un código html(ataque XSS) y probamos en las distintas interfaces.

En este caso, tu entrada se refleja entre `<script> [...] </script>` etiquetas de una página HTML, dentro de un archivo `.js` o dentro de un atributo usando el protocolo `javascript:`:

- Si se refleja entre `<script> [...] </script>` etiquetas, incluso si tu entrada está dentro de cualquier tipo de comillas, puedes intentar inyectar `</script>` y escapar de este contexto. Esto funciona porque el **navegador primero analizará las etiquetas HTML** y luego el contenido, por lo tanto, no notará que tu etiqueta inyectada `</script>` está dentro del código HTML.
- Si se refleja **dentro de una cadena JS** y el último truco no está funcionando, necesitarías **salir** de la cadena, **ejecutar** tu código y **reconstruir** el código JS (si hay algún error, no se ejecutará):

An easy way to find out if a field is vulnerable to an XSS attack is to use the `alert()` or `console.log()` methods. See one of them to find out which field is vulnerable.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	\$0.00
Dynex - Traditional Notebook Case	27.99	1	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$0.00

Enter your credit card number:

`<script> alert() </script>`

Enter your three digit access code:

Purchase

Try again. We do want to see a specific JavaScript mentioned in the goal of the assignment (in case you are trying to do something fancier).

Thank you for shopping at WebGoat.

Your support is appreciated

We have charged credit card:

\$1997.96

- En los primeros intentos da error y nos devuelve el localhost.

Shopping Cart Items -- To Buy Now

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	\$0.00
Dynex - Traditional Notebook Case	27.99	1	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$0.00

Enter your credit card number:

`<script>`

Enter your three digit access code:

Purchase

again. We do want to see a specific JavaScript mentioned in the goal of the assignment (in case you are trying to do something fancier).

Thank you for shopping at WebGoat.

Your support is appreciated

We have charged credit card:

97.96

localhost:8080

OK

- Tras poner el script por encima del número de tarjeta que presentaba la primera interfaz el script es inyectado con éxito.

field is vulnerable.

Shopping Cart

Shopping Cart Items -- To Buy Now	Price	Quantity	Total
Studio RTA - Laptop/Reading Cart with Tilting Surface - Cherry	69.99	1	\$0.00
Dynex - Traditional Notebook Case	27.99	1	\$0.00
Hewlett-Packard - Pavilion Notebook with Intel Centrino	1599.99	1	\$0.00
3 - Year Performance Service Plan \$1000 and Over	299.99	1	\$0.00

Enter your credit card number:

Enter your three digit access code:

Purchase

Congratulations, but alerts are not very impressive are they? Let's continue to the next assignment.

Thank you for shopping at WebGoat.

Your support is appreciated

We have charged credit card:4128 3214 0002 1999

\$1997.96

A5 - Security

Misconfiguration

- **XXE Injection**

- Se nos presenta una imagen y un apartado donde escribir simulando un chat.
- Abrimos el BurpSuite donde activaremos un proxy para interceptar la petición y poder realizar el ataque XXE.

Forgery >

>

>

>



john doe uploaded a photo.
24 days ago



&xxe;

Submit



david1 2024-12-07, 22:38:28
sadafadf



webgoat 2024-12-07, 22:24:48
Silly cat....



guest 2024-12-07, 22:24:48
I think I will use this picture in one of my projects.



guest 2024-12-07, 22:24:48
Lol!! :-).

- En este caso el ataque se realizará mediante la petición HOST de BurpSuite, donde aparece el código XML.

Intercept on

Forward

Drop

Time	Type	Direction	Method	URL
16:38:53 7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc
16:38:53 7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc
16:38:54 7 D...	HTTP	→ Request	POST	http://localhost:8080/WebGoat/xxe/simple
16:39:18 7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc
16:39:18 7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc
16:39:23 7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc

Request

Pretty

Raw

Hex

8

X-Requested-With: XMLHttpRequest

9

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.6723.1

10

Accept: */*

11

Content-Type: application/xml

12

Origin: http://localhost:8080

13

Sec-Fetch-Site: same-origin

14

Sec-Fetch-Mode: cors

15

Sec-Fetch-Dest: empty

16

Referer: http://localhost:8080/WebGoat/start.mvc?username=david1

17

Accept-Encoding: gzip, deflate, br

18

Cookie: JSESSIONID=sUGr7L7isXkqhykJtdDYi-VLqNmzqRhyR_8mwLqZ

19

Connection: keep-alive

20

21

<?xml version="1.0"?><comment> <text>&xxe;</text></comment>

?

⚙

←

→

Search

Event log (1)

All issues

- Modificando el código de esta misma petición es donde podremos comenzar el ataque XXE.

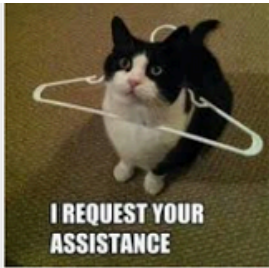
Time	Type	Direction	Method	URL
17:22:57.7 D...	HTTP	→ Request	POST	http://localhost:8080/WebGoat/xxe/simple
17:22:58.7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc
17:22:58.7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc
17:23:23.7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonoverview.mvc
17:23:23.7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc
17:23:28.7 D...	HTTP	→ Request	GET	http://localhost:8080/WebGoat/service/lessonmenu.mvc

```

Request
Pretty Raw Hex
18 Cookie: JSESSIONID=sU6r7L7isXkqhykJtdDY1-VLqNmzqRhyR_8mwLqZ
19 Connection: keep-alive
20
21 <?xml version="1.0"?>
22 <!DOCTYPE another [
23   <ENTITY fs SYSTEM "file:///">
24 ]>
25 <comment>
26   <text>
27     Hello
28     &fs:
29   </text>
30 </comment>
31

```

- Hecho esto, se le da a “Forward” en el proxy de BurpSuite para que muestre el resultado en la página.



Add a comment

Submit



david1 2024-12-07, 23:21:12

Hello __cacert_entrypoint.sh .dockerenv bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp
usr var



webgoat 2024-12-07, 23:17:18

Silly cat....



guest 2024-12-07, 23:17:18

I think I will use this picture in one of my projects.



guest 2024-12-07, 23:17:18

Lol!! :-).

- La página muestra el mensaje de “Congratulations” que significa que el ataque se ha realizado con éxito.



Add a comment

Submit



david1 2024-12-07, 23:17:58



webgoat 2024-12-07, 23:17:18

Silly cat....



guest 2024-12-07, 23:17:18

I think I will use this picture in one of my projects.



guest 2024-12-07, 23:17:18

Lol!! :-).

Congratulations. You have successfully completed the assignment.

A6 - Vuln & outdated Components

Vulnerable Components

[Reset lesson](#)

← 1 2 3 4 5 6 7 8 9 10 11 12 13 →

The exploit is not always in "your" code

Below is an example of using the same WebGoat source code, but different versions of the jquery-ui component. One is exploitable; one is not.

jquery-ui:1.10.4

This example allows the user to specify the content of the "closeText" for the jquery-ui dialog. This is an unlikely development scenario, however the jquery-ui dialog (TBD - show exploit link) does not defend against XSS in the button text of the close dialog.

Clicking go will execute a jquery-ui close dialog:

This dialog should have exploited a known flaw in jquery-ui:1.10.4 and allowed a XSS attack to occur

jquery-ui:1.12.0 Not Vulnerable

Using the same WebGoat source code but upgrading the jquery-ui library to a non-vulnerable version eliminates the exploit.

Clicking go will execute a jquery-ui close dialog:

- Nos muestra la diferencia de utilizar un mismo exploit en distintas versiones de una misma aplicación, ya que estas se corrigen con las actualizaciones de la versión o por el contrario se encuentran fallas de seguridad en versiones más modernas debido a no tapar estas vulnerabilidades.

A7 – Identity & Auth Failure

- **Secure Passwords**

- Tenemos que conseguir una contraseña segura, esto se consigue intercalando mayúsculas con minúsculas, número, además, de letras no tan comunes como la “ñ” que aporta seguridad extra.

How long could it take to brute force your password?

In this assignment, you have to type in a password that is strong enough (at least 4/4).

After you finish this assignment we highly recommend you try some passwords below to see why they are not good choices:

- password
- johnsmith
- 2018/10/4
- 1992home
- abcabc
- fffget
- poiuz
- @dmin

☐ Show password

You have succeeded! The password is secure enough.

Your Password: *****

Length: 18

Estimated guesses needed to crack your password: 14400000000010000

Score: 4/4

Estimated cracking time: 45662100 years 166 days 16 hours 16 minutes 40 seconds

Score: 4/4

¿Cómo se podría reforzar la seguridad?

- **Hardening**

- Medidas para inyección SQL.

Usar consultas parametrizadas o prepared statements:
Esto asegura que las entradas del usuario no se ejecuten como código SQL.

Ejemplo en Java (JDBC):

```
String query = "SELECT * FROM users WHERE username =  
? AND password = ?";  
PreparedStatement pstmt =  
connection.prepareStatement(query);  
pstmt.setString(1, username);  
pstmt.setString(2, password);  
ResultSet rs = pstmt.executeQuery();
```

- **Validar y sanitizar entradas:**

Restringir caracteres especiales y validar formatos como correos electrónicos o números.

- **Principio de mínimos privilegios:**

Configurar las cuentas de base de datos con acceso restringido únicamente a las operaciones necesarias.

- **Medidas para ataques XML External Entity (XXE)**

Deshabilitar el procesamiento de entidades externas:

Configurar el parser XML para bloquear el acceso a DTDs y entidades externas.

Ejemplo en Java (JAXP):

```
DocumentBuilderFactory factory =  
DocumentBuilderFactory.newInstance();  
factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);  
factory.setFeature("http://xml.org/sax/features/external-general-entities", false);  
factory.setFeature("http://xml.org/sax/features/external-parameter-entities", false);
```

- **Validar y restringir el contenido XML:**

Asegúrate de procesar solo XML proveniente de fuentes confiables.

- **Usar bibliotecas modernas y seguras:**

Opta por procesadores XML que deshabiliten las entidades externas por defecto.

- **Medidas para Cross-Site Scripting (XSS)**

Escapar y sanitizar salidas:

Escapa caracteres especiales como <, >, ", y ' antes de incluirlos en HTML.

Ejemplo en JavaScript (con DOMPurify):

```
const input = "<script>alert('XSS')</script>";
const sanitizedInput = DOMPurify.sanitize(input);
document.getElementById("output").innerHTML =
sanitizedInput;
```


- **Usar Content Security Policy (CSP):**

Configura CSP en el servidor para bloquear la ejecución de scripts no autorizados.

Ejemplo en HTTP Header:

```
Content-Security-Policy: default-src 'self'; script-src 'self'
```

Webgrafía

- <https://github.com/WebGoat/WebGoat>
- <https://owasp.org/www-project-webgoat/>
- <https://docs.cycubix.com/>
- <https://es.wikipedia.org/>
- <https://chatgpt.com/>