

INFORME DE AUDITORIA WEB GOAT

ALEXANDRA OANE

INDICE

1. Ámbito y alcance de la auditoría.....	1
1.1 Introducción.....	2
2. Informe ejecutivo.....	3
2.1. A1 SQL Injection – Apartado 10.....	3
2.2. A1 SQL Injection - Apartado 11.....	5
2.3. Intenta obtener toda la información que puedas de la base de datos utilizando los fallos disponibles en la sección A1 SQL Injection.....	7
2.4. A5 Insecure Direct Object References - Apartado 3.....	7
2.5. A5 Insecure Direct Object References - Apartado 4.....	10
2.6. A5 Insecure Direct Object References - Apartado 5.....	12
2.7. A5 Missing Function Level Access Control - Apartado 2.....	14
2.8. A5 Missing Function Level Access Control - Apartado 3.....	14
2.9. A7 Cross Site Scripting - Apartado - Apartado 7.....	17
3. Descripción del proceso de auditoría.....	17
3.1. Reconocimiento/Information gathering.....	18
3.2. Explotación de vulnerabilidades detectadas.....	19
3.3. Post-explotación.....	20
3.4. Posibles mitigaciones	24
3.5. Herramientas utilizadas	24

1. ÁMBITO Y ALCANCE DE LA AUDITORÍA

1.1. INTRODUCCIÓN

La presente auditoria se ha realizado para la aplicación web “WebGoat” (<http://127.0.0.1:8080/WebGoat>).

Inicialmente se realizó un registro del usuario para poder acceder. A continuación se realizó la búsqueda de los puertos abiertos, el sistema operativo y los lenguajes de programación empleados en la aplicación web.

Los puertos abiertos: El puerto es 80.

El sistema operativo: No he podido encontrar el sistema operativo.

Lenguajes de programación: el lenguaje de programación es javascript.

Al acceder a WebGoat, se accedió a las distintas secciones situadas en la página principal. A continuación se realizó la descarga e instalación de ZAP, una de las herramientas empleadas para realizar la auditoria. También se ha usado nmap,

Además durante la realización del informe, se ha buscado información de Sql Injection para poder realizar los apartados. Los apartados opcionales han sido necesario resolverlos para poder obtener las soluciones de los ejercicios posteriores en algunos de los casos.

2. INFORME EJECUTIVO

2.1. A1 SQL Injection – Apartado 10

2.1.1. Proceso realizado

Al inicio del ejercicio se observa que es necesario resolver apartados anteriores para poder obtener la solución.

En este apartado se debe averiguar Login_Count y User_id. Para poder acceder se realizamo una búsqueda de información sobre Numeric SQL Injection. Tras intentar la consulta con los distintos campos y probando distintos datos finalmente introduciendo en los campos:

- Login_Count= 0
- User_id= 0 OR 1+1

Try It! Numeric SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query by concatenating a number making it susceptible to Numeric SQL injection:

```
"SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND userid = " + User_ID;
```

Using the two Input Fields below, try to retrieve all the data from the users table.

Warning: Only one of these fields is susceptible to SQL Injection. You need to find out which, to successfully retrieve all the data.

✓

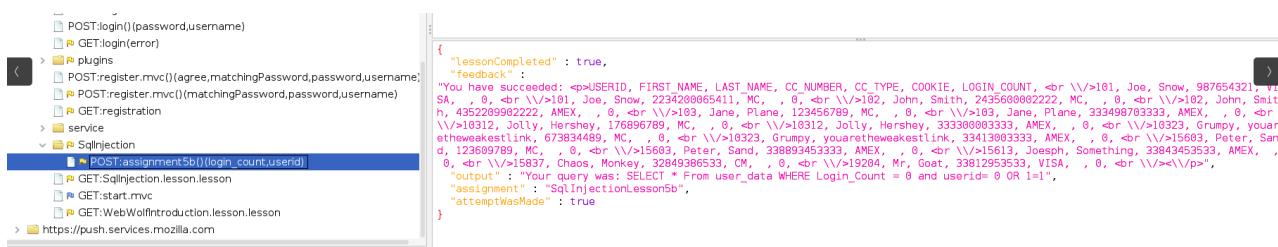
Login_Count:

User_Id:

You have succeeded:

```
USERID,FIRST_NAME,LAST_NAME,CC_NUMBER,CC_TYPE,COOKIE,LOGIN_COUNT,  
101,Joe,Snow,987654321,VISA,,0,  
101,Joe,Snow,2234200065411,MC,,0,  
102,John,Smith,2435600002222,MC,,0,  
102,John,Smith,4352209902222,AMEX,,0,  
103,Jane,Plane,123456789,MC,,0,  
103,Jane,Plane,333498703333,AMEX,,0,  
10312,Jolly,Hershey,176896789,MC,,0,  
10312,Jolly,Hershey,333300003333,AMEX,,0,  
10323,Grumpy,youaretheweakestlink,673834489,MC,,0,  
10323,Grumpy,youaretheweakestlink,33413003333,AMEX,,0,  
15603,Peter,Sand,123609789,MC,,0,  
15603,Peter,Sand,338893453333,AMEX,,0,  
15613,Joesh,Something,33843453533,AMEX,,0,  
15837,Chaos,Monkey,32849386533,CM,,0,  
19204,Mr,Goat,33812953533,VISA,,0,
```

Your query was: SELECT * From user_data WHERE Login_Count = 0 and userid= 0 OR 1+1



Se observa que introduciendo una segunda opción con los siguientes datos:

- Login_Count= 1
- User_id= 1 OR '1' = '1'

Try It! Numeric SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query by concatenating a number making it susceptible to Numeric SQL injection:

```
"SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND userid = " + User_ID;
```

Using the two Input Fields below, try to retrieve all the data from the users table.

Warning: Only one of these fields is susceptible to SQL Injection. You need to find out which, to successfully retrieve all the data.

✓

Login_Count:

User_Id:

You have succeeded:

```
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,
```

Your query was: SELECT * From user_data WHERE Login_Count = 1 and userid= 1 OR '1' = '1'

HTTP/1.1 200 OK

Connection: keep-alive

X-XSS-Protection: 1; mode=block

X-Content-Type-Options: nosniff

X-Frame-Options: DENY

Content-Type: application/json

Date: Sat, 09 Jul 2022 16:05:39 GMT

```
{
  "lessonCompleted" : true,
  "feedback" :
  "You have succeeded: <p>USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN COUNT, <br >\n101, Joe, Snow, 987654321, VISA, , 0, <br >\n101, Joe, Snow, 2234200065411, MC, , 0, <br >\n102, John, Smith, 2435600002222, MC, , 0, <br >\n102, John, Smith, 4352209902222, AMEX, , 0,\n<br >\n103, Jane, Plane, 123456789, MC, , 0, <br >\n103, Jane, Plane, 333498703333, AMEX, , 0, <br >\n10312, Jolly, Hershey, 176896789, MC, , 0, <br >\n10312, Jolly, Hershey, 333300003333, AMEX, , 0, <br >\n10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0, <br >\n10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0, <br >\n15603, Peter, Sand, 123609789, MC, , 0, <br >\n15603, Peter, Sand, 338893453333, AMEX, , 0, <br >\n15613, Joesph, Something, 33843453533, AMEX, , 0, <br >\n15837, Chaos, Monkey, 32849386533, CM, , 0, <br >\n19204, Mr, Goat, 33812953533, VISA, , 0, <br ></p>",
  "output" : "Your query was: SELECT * From user_data WHERE Login_Count = 1 and userid= 1 OR '1' = '1'",
  "assignment" : "SqlInjectionLesson5b",
  "attemptWasMade" : true
}
```

2.1.2. Vulnerabilidades

Las vulnerabilidades que se observan en este apartado, es una serie de fallo que presenta la aplicación web, probando manualmente distintos datos en un orden determinado para poder acceder a los datos de la base de datos.

2.1.3. Conclusiones

Para poder acceder a todos los datos de los empleados presentados en la tabla en el apartado 2, se comprueba que se produce un fallo que permite acceder a los nombres completos y datos de sus tarjetas bancarias.

2.1.4. Recomendaciones

Se recomienda reforzar la seguridad de la aplicación web.

2.2. A1 SQL Injection - Apartado 11

2.2.1. Proceso realizado

En este apartado se pide realizar una cadena de inyección SQL. La forma que nos presentan de ayuda para poder acceder siguiendo el orden establecido, se han insertado los siguientes datos:

EMPLOYEE NAME: 0

Authentication TAN: 0 OR 1=1

El resultado ha sido vulnerar la seguridad y poder acceder a la información interna confidencial de los empleados, sin tener acceso.

También se intentó acceder insertando datos que se presentaban en la tabla, pero no ha sido posible. Insertando apellidos de los empleados en “employee name” y en “authentication TAN”

‘OR ‘1’ = 1’ (ejemplo), no se ha podido acceder.

Dando el valor “0” inicial y en el segundo apartado la combinación de “0” ó 1=1, es en el único caso en el que se ha producido el fallo.

What is String SQL injection?

If an application builds SQL queries simply by concatenating user supplied strings to the query, the application is likely very susceptible to String SQL injection.

More specifically, if a user supplied string simply gets concatenated to a SQL query without any sanitization or preparation, then you may be able to modify the query's behavior by simply inserting quotation marks into an input field example, you could end the string parameter with quotation marks and input your own SQL after that.

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:

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

```
HTTP/1.1 200 OK
Connection: keep-alive
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Sat, 09 Jul 2022 16:00:44 GMT
```

```
{
    "lessonCompleted" : true,
    "feedback" :
    "You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!",
    "output" :
    "<table><r><th>USERID<\th><th>FIRST_NAME<\th><th>LAST_NAME<\th><th>DEPARTMENT<\th><th>SALARY<\th><th>AUTH_TAN<\th><\tr>
    <r><d>32147</d><d>Pauline</d><d>Travers</d><d>Accounting</d><d>46000</d><d>P45JSI</d><\tr><d>34477</d>
    <d>Abraham</d><d>Holman</d><d>Development</d><d>50000</d><d>UU2ALK</d><\tr><d>37648</d><d>John</d>
    <d>Smith</d><d>Marketing</d><d>64350</d><d>3SL99A</d><\tr><r><d>89762</d><d>Tobi</d><d>Barnett</d>
    <d>Development</d><d>77000</d><d>TA9LL1</d><\tr><r><d>96134</d><d>Bob</d><d>Franco</d><d>Marketing</d>
    <d>83700</d><d>L09S2V</d><\tr><\table>",
    "assignment" : "SqlInjectionLesson8",
    "attemptWasMade" : true
}
```

Timestamps

Rules

Download



2.2.2. Vulnerabilidades

En este apartado se observa que al insertar la combinación numérica de 0 y 1, dando estos valores se produce un fallo que permite acceder a datos confidenciales de los empleados.

2.2.3. Conclusiones

Se ha podido acceder a los datos confidenciales de la base de datos, de los empleados. Este fallo se produce al realizar una cadena de inyección SQL, introduciendo los valores numéricos 0 y 1.

2.2.4. Recomendaciones

Se recomienda reforzar la seguridad.

2.3. Intenta obtener toda la información que puedas de la base de datos utilizando los fallos disponibles en la sección A1 SQL Injection.

2.3.1. Proceso realizado

Se ha tratado de realizar la búsqueda de toda la información posible de la base de datos usando los fallos posibles y no ha sido posible.

2.3.2. Vulnerabilidades

2.3.3. Conclusiones

2.3.4. Recomendaciones

2.4. A5 Insecure Direct Object References - Apartado 2 y Apartado 3

2.4.1. Proceso realizado

En este apartado, se retrocedió al ejercicio anterior para introducir los datos en los campos que se señalan. Dentro de la aplicación web de WebGoat se realizó la inspección a través de “more tools”. Se observa en el campo de “response” los datos necesarios para poder realizar el apartado 3. Los dos atributos que se solicitan en el ejercicio pero no están presentes, se pueden obtener accediendo al campo anteriormente mencionado.

Además también se ha podido observar que también se muestra el userID.

Insecure Direct Object References



Show hints Reset lesson

1 2 3 4 5 6

Authenticate First, Abuse Authorization Later

Many access control issues are susceptible to attack from an authenticated-but-unauthorized user. So, let's start by legitimately authenticating. Then, we will look for ways to bypass or abuse Authorization.

The id and password for the account in this case are 'tom' and 'cat' (It is an insecure app, right?).

After authenticating, proceed to the next screen.

user/pass user: pass: Submit

Debugger Network Style Editor Performance Memory Storage Accessibility Application

File Initiator Type Transferred Size

27.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	8.01 KB	7.98 KB
27.0.0.1:8080	profile	jquery.min.js:2 (xhr)	json	143 B	104 B
27.0.0.1:8080	lessonoverview.mvc	jquery.min.js:2 (xhr)	json	809 B	770 B
27.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	8.01 KB	7.98 KB
27.0.0.1:8080	lessonoverview.mvc	jquery.min.js:2 (xhr)	json	809 B	770 B
27.0.0.1:8080	lessonmenu.mvc	jquery.min.js:2 (xhr)	json	8.01 KB	7.98 KB

0.94 KB transferred Finish: 4.84 min

Headers Cookies Request Response Timings StackTrace Security

JSON

role: 3
color: "yellow"
size: "small"
name: "Tom Cat"
userId: "2342384"

Insecure Direct Object References



Show hints Reset lesson

1 2 3 4 5 6 +

Observing Differences & Behaviors

A consistent principle from the offensive side of AppSec is to view differences from the raw response to what is visible. In other words (as you may have already noted in the client-side filtering lesson), there is often data in the raw response that doesn't show up on the screen/page. View the profile below and take note of the differences.

View Profile
name:Tom Cat
color:yellow
size:small

In the text input below, list the two attributes that are in the server's response, but don't show above in the profile.
 Submit Diffs
Correct, the two attributes not displayed are userid & role. Keep those in mind

2.4.2. Vulnerabilidades

Las vulnerabilidades que se observan en este apartado, sin la necesidad de abrir “Inspector”, se puede averiguar. Pero con la ayuda de la herramienta se ha podido acceder a los siguientes apartados. Obteniendo los atributos de señalan en el ejercicio.

2.4.3. Conclusiones

Con la ayuda de la herramienta de desarrollador “Inspector”, buscando entre las distintas secciones se ha podido encontrar desde Network y Response, los dos atributos solicitados.

2.4.4. Recomendaciones

Se recomienda reforzar la seguridad de las referencias de los objetos.

2.5. A5 Insecure Direct Object References - Apartado 4

2.5.1. Proceso realizado

En este apartado con la obtención de los datos de los apartados anteriores en la dirección que se solicita se obtiene en el apartado de “Headers”.

webgoat/IDOR/profile/numero del userid que en este caso es 2342384

injection → developer tools → network

En la sección de network y en el apartado response se puede observar los datos que se muestran.

The screenshot shows the Network tab of a browser's developer tools. A red box highlights a JSON response from the URL 'WebGoat/profile'. The response contains the following data:

```
HTTP/1.1 200 OK
Connection: keep-alive
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Content-Type: application/json
Date: Sat, 09 Jul 2022 16:52:49 GMT

{
  "lessonCompleted": true,
  "feedback": "Congratulations, you have used the alternate Url\\route to view your own profile.",
  "output": "{role=3, color=yellow, size=small, name=Tom Cat, userId=2342384}",
  "assignment": "IDORviewOwnProfileAltUrl",
  "attemptWasMade": true
}
```

The 'Response' tab is selected in the Network tool. The JSON content is displayed in pink, indicating it is a JSON object. The 'Headers' tab shows the HTTP headers listed above.

2.5.2. Vulnerabilidades

Se ha observado a través de la obtención de los datos anteriores y con la ayuda de la herramienta, el fallo se produce por los problemas del control de acceso por un usuario autenticado pero no autorizado.

2.5.3. Conclusiones

Se han podido extraer los distintos datos a través de la herramienta de desarrollador.

2.5.4. Recomendaciones

Reforzar la seguridad, realizando configuraciones más restrictivas.

2.6. A5 Insecure Direct Object References - Apartado 5

2.6.1. Proceso realizado

En este apartado al tratar de acceder a los dos perfiles se puede observar, se ha tratado de realizar el ejercicio pero no ha sido posible averiguar la solución.

status 500 y error "Internal Server Error"

Playing with the Patterns

View Another Profile

View someone else's profile by using the alternate path you already used to view your own profile. Use the 'View Profile' button and intercept/modify the request to view another profile. Alternatively, you may also just be able to use a manual GET request with your browser.

[View Profile](#)

Edit Another Profile

Older apps may follow different patterns, but RESTful apps (which is what's going on here) often just change methods (and include a body or not) to perform different functions.

Use that knowledge to take the same base request, change its method, path and body (payload) to modify another user's (Buffalo Bill's) profile. Change the role to something lower (since higher privilege roles and users are usually lower numbers). Also change the user's color to 'red'.

[View Profile](#)

ger Network Style Editor Performance Memory Storage Accessibility Application

File	Initiator	Type	Transferred	Size
lessonoverview.mvc	jquery.min.js:2 (xhr)	json	808 B	769 B
lessonmenu.mvc	jquery.min.js:2 (xhr)	json	8.01 KB	7.98 KB
lessonoverview.mvc	jquery.min.js:2 (xhr)	json	808 B	769 B
[userid]	jquery.min.js:2 (xhr)	json	11.45 KB (raced)	11.42 KB
lessonmenu.mvc	jquery.min.js:2 (xhr)	json	8.01 KB	7.98 KB
lessonoverview.mvc	jquery.min.js:2 (xhr)	json	808 B	769 B

timestamp: "2022-07-10T10:23:24.457+00:00"
status: 500
error: "Internal Server Error"
trace: "java.lang.NullPointerException: Cannot invoke 'String equals(Object)' on a null object
at org.owasp.webgoat.lessons.idor.UserProfile.getUserid()
at org.owasp.webgoat.lessons.idor.IDORViewOtherProfile.completed(IDORViewOtherProfile.java:137)
at org.owasp.webgoat.lessons.idor.IDORViewOtherProfile.access\$000(IDORViewOtherProfile.java:25)
at org.owasp.webgoat.lessons.idor.IDORViewOtherProfile\$1.onCompleted(IDORViewOtherProfile.java:132)
at org.owasp.webgoat.lessons.idor.IDORViewOtherProfile\$1.onCompleted(IDORViewOtherProfile.java:132)"

```

HTTP/1.1 500 INTERNAL SERVER ERROR
Connection: keep-alive
Content-Type: application/json
Date: Sat, 09 Jul 2022 16:58:23 GMT

{
  "timestamp": "2022-07-09T16:58:23.977+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "trace": [
    "java.lang.NullPointerException: Cannot invoke \"String.equals(Object)\" because the return value of \"org.owasp.wer.UserProfile.getId()\" is null\\n\\tat org.owasp.webgoat.lessons.idor.IDORViewOtherProfile.completed(IDORViewOtherProfile.java:50)\\n\\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)\\n\\tat java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)\\n\\tat java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\\n\\tat java.lang.reflect.Method.invoke(Method.java:568)\\n\\tat org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:205)\\n\\tat org.springframework.web.method.support.InvocableHandlerMethod.invokeAndHandle(InvocableHandlerMethod.java:17)\\n\\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:895)\\n\\tat org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:808)\\n\\tat org.springframework.web.servlet.mvc.method.AbstractHandlerAdapter.handle(AbstractHandlerAdapter.java:87)\\n\\tat org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1040)\\n\\tat org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:941)\\n\\tat org.springframework.web.servlet.FrameworkServlet.processEvent(FrameworkServlet.java:1008)\\n\\tat org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:892)\\n\\tat javax.servlet.http.HttpServlet.service(HttpServlet.java:682)\\n\\tat org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:876)\\n\\tat javax.servlet.http.HttpServlet.service(HttpServlet.java:770)\\n\\tat org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:423)\\n\\tat org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65)\\n\\tat org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol.java:622)\\n\\tat org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.lambda$7(NioEndpoint.java:1607)\\n\\tat org.apache.tomcat.util.net.NioEndpoint$SocketProcessor$$Lambda$137/133744328.run(Unknown Source)\\n\\tat java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)\\n\\tat java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)\\n\\tat org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)\\n\\tat java.lang.Thread.run(Thread.java:748)"
  ]
}

```

Alerts Output WebSockets AJAX Spider +

2.6.2. Vulnerabilidades

2.6.3. Conclusiones

2.6.4. Recomendaciones

2.7. A5 Missing Function Level Access Control - Apartado 2

2.7.1. Proceso realizado

En este apartado se pide averiguar dos objetos que se encuentran ocultos. Tras realizar varios intentos de manera aleatoria, los distintos resultados eran incorrectos.

Se accedió a través de “Inspector” y se realizó una búsqueda, en una de las secciones se pudo acceder a la parte de admin y dar la visibilidad a los campos ocultos.

Al lado de “Messages” se puede comprobar que se encuentra el campo de “Administrador”.

The screenshot shows the Chrome DevTools Elements tab with the DOM tree open. A dropdown menu under 'Admin' is expanded, displaying three items: 'Users', 'Users', and 'Config'. The 'hidden-menu-item' class is applied to the first two items. The right sidebar displays the CSS styles for these elements, including the dropdown menu itself and its items.

The screenshot shows the Chrome DevTools Elements tab with the DOM tree open. The 'Admin' dropdown is expanded, showing 'Users', 'Users', and 'Config'. The 'hidden-menu-item' class is applied to the first two items. The right sidebar shows the CSS styles for the dropdown menu and its items.

2.7.2. Vulnerabilidades

Las vulnerabilidades que se observan en este apartado debido a un error que presenta, es poder acceder a la visibilidad de los campos ocultos. Se obtiene de una manera sencilla buscando en los datos de la consola.

2.7.3. Conclusiones

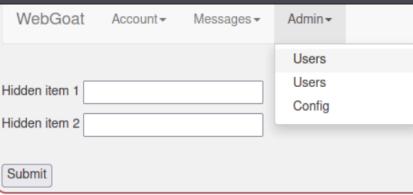
Se ha podido acceder a los campos ocultos con la ayuda de las herramientas de desarrollador.

2.7.4. Recomendaciones

Se recomienda reforzar la visibilidad de los campos que se quieren ocultar a los posibles atacantes.

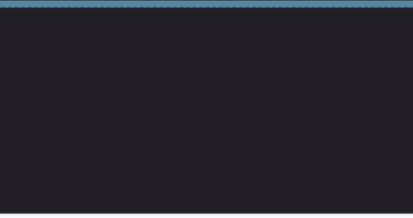
2.8. A5 Missing Function Level Access Control - Apartado 3

2.8.1. Proceso realizado



The screenshot shows a browser window for the WebGoat application. At the top, there are navigation links: 'WebGoat', 'Account', 'Messages', and 'Admin'. The 'Admin' link is currently selected. A dropdown menu is open under 'Admin', showing 'Users' and 'Config' as options. Below the dropdown, there are two input fields with placeholder text 'Hidden item 1' and 'Hidden item 2'. At the bottom of the form is a 'Submit' button. The URL in the address bar is <https://127.0.0.1:8080/access-control/users>. The browser's developer tools are open, specifically the 'Inspector' tab, which displays the HTML structure of the page. The 'Elements' panel shows the DOM tree, and the 'Style' panel on the right shows CSS rules applied to the elements. The 'Users' and 'Config' menu items are highlighted in blue in the DOM tree.

En el siguiente apartado, se solicita que se encuentre dos objetos que no se muestran. En un primer momento se realizaron pruebas con distintos datos, al no dar resultado, se usó “Inspector” y se fue abriendo y comprobando cada uno de los resultados. En este apartado no se ha podido encontrar el Hash.



The screenshot shows the browser's developer tools with the 'Elements' panel active. The 'Inherited from html' section is expanded, showing a CSS rule for 'color' with a value of 'rgb(251, 251, 254)'. This indicates that the 'color' property is being inherited from the 'html' element. The 'Style' panel on the right shows other CSS rules, such as 'background: 0 0' and 'background-color: rgba(0, 0, 0, 0);'. The URL in the address bar is <https://127.0.0.1:8080/>.

Se comprueba con la siguiente opción control users admin, en este caso también se observa que no hay ningún dato que se proporcione.

Challenges > Your mission

Find two invisible menu items in the menu below that are or would be of interest to an attacker/malicious user and submit the labels for those menu items (there are no links right now in the menu).

WebGoat Account ▾ Messages ▾ Admin ▾

Hidden item 1

Hidden item 2

Users
Users
Config

<https://127.0.0.1:8080/access-control/users-admin-fix>

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Search HTML

```
<a href="#">Compose Message</a>
</li>
</li>
<li class="hidden-menu-item dropdown open">
  <a class="dropdown-toggle" href="#" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">&lt;/a> [event]
  <ul class="dropdown-menu" aria-labelledby="admin">
    <li>
      <a href="/access-control/users">Users</a>
    </li>
    <li>
      <a href="/access-control/users-admin-fix">Users</a>
    </li>
    <li>
      <a href="/access-control/config">Config</a>
    </li>
  </ul>
</li>
```

Filter Styles :hover .cls

Inherited from ul

.nav { list-style-type: none; }

Inherited from body

body { font-family: "Helvetica Neue", Helvetica, Arial, sans-serif; font-size: 14px; line-height: 1.42857143; color: #333; }

body { color: #555555; font-family: "Open Sans", sans-serif; }

wrapper > div.attack-container > nav.navbar.navbar-default > div.container-fluid > div#alignment-example.collapse.navbar-collapse > ul.nav.navbar-nav > li.hidden-menu-item.dropdown.open > ul.dropdown-menu > li

Filter Output

initialize goat app router

Search HTML

```
<html>
  <head>
    <link rel="stylesheet" href="resource://content-accessible/plaintext.css"></link>
  </head>
  <body>
    <pre></pre>
  </body>
</html>
```

ml > body > pre

GET https://127.0.0.1:8080/favicon.ico

Filter Styles :hover .cls

element { }

:root { plaintext.css:22 @prefers-color-scheme: dark; }
background: #34, 27, 30;
color: #251, 254, 253;

background-attachment: scroll;
background-clip: border-box;
background-color: #34, 27, 30;
background-image: none;
background-origin: padding-box;
background-position: 0% 0%;
background-repeat: repeat;
background-size: 100% 100%;

Layout Comp

Filter Styles

inline

Errors Warnings Logs Info Dev

Y también se comprueba con la última opción, configuración. En este caso tampoco se observa ningún dato que permita acceder.

Your mission

Find two invisible menu items in the menu below that are or would be of interest to an attacker/malicious user and submit the labels for those menu items (there are no links right now in the menus).

The screenshot shows a browser window with a navigation bar at the top. A dropdown menu is open under the 'Admin' button. The menu contains three items: 'Users', 'Users', and 'Config'. The first two items ('Users') are highlighted with red boxes, indicating they are the two invisible menu items to find.

The screenshot shows a browser developer tools interface. The left panel displays the DOM structure of the dropdown menu. The 'Users' item is selected in the list. The right panel shows the CSS styles applied to this element, which are identical to the styles for the other 'Users' item in the menu.

The screenshot shows a browser developer tools interface. The left panel displays the page source code. The 'link' element in the head section is selected. The right panel shows the CSS styles applied to this element, which are identical to the styles for the other 'link' element in the page.

2.8.2. Vulnerabilidades

2.8.3. Conclusiones

Se ha tratado de realizar toda la búsqueda e inspección pero no ha sido posible averiguar el Hash.

2.8.4. Recomendaciones

2.9. A7 Cross Site Scripting - Apartado - Apartado 7

2.9.1. Proceso realizado

No se ha podido realizar el apartado.

2.9.2. Vulnerabilidades

2.9.3. Conclusiones

2.9.4. Recomendaciones

3. DESCRIPCIÓN DEL PROCESO DE AUDITORIA

3.1. Reconocimiento/Information gathering

Se ha monitorizado y rastreado toda la posible información para poder acceder a la información confidencial y la posibilidad de vulnerar la seguridad e insertar datos.

En los distintos apartados accediendo a la herramientas de developer y con la ayuda de ZAP se pueden realizar algunos de los ejercicios.

3.2. Explotación de vulnerabilidades detectadas

Durante el proceso de reconocimiento se han detectado inicialmente 14 alertas, de las cuales 4 son de tipo medio y 10 son de tipo bajo.

- ▼ Alerts (14)
 - Absence of Anti-CSRF Tokens (12)
 - Content Security Policy (CSP) Header Not Set
 - Cross-Domain Misconfiguration (28)
 - Vulnerable JS Library (5)
 - Absence of Anti-CSRF Tokens
 - Cookie No HttpOnly Flag (2)
 - Cookie without SameSite Attribute (2)
 - Incomplete or No Cache-control Header Set (2)
 - Timestamp Disclosure - Unix (232)
 - X-Content-Type-Options Header Missing (21)
 - Charset Mismatch

CSP: Wildcard Directive		
URL:	http://127.0.0.1:37903/	
Risk:	Medium	
Confidence:	Medium	
Parameter:		
Attack:		
Evidence:	default-src 'none'; script-src 'self'; connect-src 'self'; child-src 'self'; img-src 'self' data:; font-src 'self' data:; style-src 'self'	
CWE ID:	693	
WASC ID:	15	
Source:	Passive (10055 - CSP)	
Description:	frame-ancestors, form-action	
The directive(s):	frame-ancestors, form-action are among the directives that do not fallback to default-src, missing/excluding them is the same as allowing anything.	
Other Info:		
Solution:	Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.	
Reference:	http://www.w3.org/TR/CSP2/ http://www.w3.org/TR/CSP/ http://caniuse.com/#search=content+security+policy	
Alert Tags:		
	Key	Value
OWASP_2021_A05		https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
OWASP_2017_A06		https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfigur

Cross-Domain Misconfiguration		
URL:	https://firefox.settings.services.mozilla.com/v1/buckets/monitor/collections/changes/changeset?collection=partitioning-exempt-urls&bucket=main&_expected=0	
Risk:	Medium	
Confidence:	Medium	
Parameter:		
Attack:		
Evidence:	Access-Control-Allow-Origin: *	
CWE ID:	264	
WASC ID:	14	
Source:	Passive (10098 - Cross-Domain Misconfiguration)	
Description:	Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server	
Other Info:	The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.	
Solution:	Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Policy (SOP) in a more restrictive manner.	
Reference:	https://vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy	
Alert Tags:		
	Key	Value
OWASP_2021_A01		https://owasp.org/Top10/A01_2021-Broken_Access_Control/
OWASP_2017_A05		https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.htm

Vulnerable JS Library

URL: http://127.0.0.1:8080/WebGoat/js/libs/jquery.min.js
Risk: ⚠️ Medium
Confidence: Medium
Parameter:
Attack:
Evidence: /*! jQuery v3.4.1
CWE ID: 829
WASC ID:
Source: Passive (10003 - Vulnerable JS Library)
Description:
The identified library jquery, version 3.4.1 is vulnerable.

Other Info:

CVE-2020-11023
CVE-2020-11022

Solution:

Please upgrade to the latest version of jquery.

Reference:

<https://blog.jquery.com/2020/04/10/jquery-3-5-0-released/>

Alert Tags:

Key	Value
OWASP_2017_A09	https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Kno
OWASP_2021_A06	https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

Absence of Anti-CSRF Tokens

URL: http://127.0.0.1:8080/WebGoat/login
Risk: ⚠️ Low
Confidence: Medium
Parameter:
Attack:
Evidence: <form action="/WebGoat/login" method='POST' style="width: 200px;">
CWE ID: 352
WASC ID: 9
Source: Passive (10202 - Absence of Anti-CSRF Tokens)

Description:

No Anti-CSRF tokens were found in a HTML submission form.
A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that

Other Info:

No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anonscsrf, csrf_token, _csrf, __csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "exampleInputEmail1""exampleInputPassword1"].

Solution:

Phase: Architecture and Design
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Reference:

<http://projects.webappsec.org/Cross-Site-Request-Forgery>
<http://cwe.mitre.org/data/definitions/352.html>

Alert Tags:

Key	Value
OWASP_2021_A01	https://owasp.org/Top10/A01_2021-Broken_Access_Control/
WSTG-v42-SESS-05	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Applications/B1-Broken_Access_Control/
OWASP 2017 A05	https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control/

Application Error Disclosure	
URL:	http://127.0.0.1:8080/WebGoat/IDOR/profile/alt-path
Risk:	Low
Confidence:	Medium
Parameter:	
Attack:	
Evidence:	HTTP/1.1 500 Internal Server Error
CWE ID:	200
WASC ID:	13
Source:	Passive (90022 - Application Error Disclosure)
Description:	This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information could be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.
Other Info:	
Solution:	Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client so that it can report the details on the server side and not exposing them to the user.
Reference:	
Alert Tags:	
Key	Value
WSTG-v42-ERRH-02	https://owasp.org/www-project-web-security-testing-guide/v42/4-Writing-Secure/4.1-Identifying-Vulnerabilities/4.1.1-Error-Handling/
WSTG-v42-ERRH-01	https://owasp.org/www-project-web-security-testing-guide/v42/4-Writing-Secure/4.1-Identifying-Vulnerabilities/4.1.2-Common-Vulnerabilities/
OWASP 2021 A05	https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

3.3. Post-explotación

Tras realizar en ZAP un escaneo se han detectado 19 alertas. De las cuales 2 son de nivel alto, 6 alertadas de nivel medio y 11 son de nivel bajo.

The screenshot shows the ZAP interface with the 'Alerts' tab selected. It displays a hierarchical list of 19 detected issues. The top-level category is 'Alerts (19)'. Underneath, several sub-categories are listed, each preceded by a red exclamation mark icon. The first item, 'Cross Site Scripting (Reflected)', is highlighted with a blue background, indicating it is the current selection or the focus of the review. Other visible categories include SQL Injection, Absence of Anti-CSRF Tokens, Content Security Policy (CSP) Header Not Set, Cross-Domain Misconfiguration, Format String Error, Parameter Tampering, Vulnerable JS Library, Absence of Anti-CSRF Tokens, Cookie No HttpOnly Flag, Cookie without SameSite Attribute, Cross Site Scripting Weakness (Reflected in JSON), Incomplete or No Cache-control Header Set, Timestamp Disclosure - Unix, X-Content-Type-Options Header Missing, Charset Mismatch, Charset Mismatch (Header Versus Meta Content-Type), Information Disclosure - Suspicious Comments, and Re-examine Cache-control Directives.

- Alerts (19)
 - > **!** Cross Site Scripting (Reflected)
 - > **!** SQL Injection (2)
 - > **!** Absence of Anti-CSRF Tokens (18)
 - > **!** Content Security Policy (CSP) Header Not Set (1)
 - > **!** Cross-Domain Misconfiguration (28)
 - > **!** Format String Error
 - > **!** Parameter Tampering (5)
 - > **!** Vulnerable JS Library (5)
 - > **!** Absence of Anti-CSRF Tokens
 - > **!** Cookie No HttpOnly Flag (4)
 - > **!** Cookie without SameSite Attribute (4)
 - > **!** Cross Site Scripting Weakness (Reflected in JSON)
 - > **!** Incomplete or No Cache-control Header Set (2)
 - > **!** Timestamp Disclosure - Unix (232)
 - > **!** X-Content-Type-Options Header Missing (21)
 - > **!** Charset Mismatch
 - > **!** Charset Mismatch (Header Versus Meta Content-Type)
 - > **!** Information Disclosure - Suspicious Comments
 - > **!** Re-examine Cache-control Directives (28)

Alertas de nivel alto.

Cross Site Scripting (Reflected)

URL: http://127.0.0.1:8080/WebGoat/CrossSiteScripting/attack5a?QTY1=1&QTY2=1&QTY3=1&QTY4=1&field1=%3Cimg+src%3Dx+onerror%3Dprompt%28%29%3E&field2=111

Risk: High

Confidence: Medium

Parameter: field1

Attack:

Evidence:

CWE ID: 79

WASC ID: 8

Source: Active (40012 - Cross Site Scripting (Reflected))

Description:

Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology.

Other Info:

Solution:

Phase: Architecture and Design

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.

Reference:

<http://projects.webappsec.org/Cross-Site-Scripting>

<http://cwe.mitre.org/data/definitions/79.html>

Alert Tags:

Key	Value
OWASP_2021_A03	https://owasp.org/Top10/A03_2021-Injection/
WSTG-v42-INPV-01	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_...
OWASP_2017_A07	https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS).html

SQL Injection

URL: http://127.0.0.1:8080/WebGoat/access-control/user-hash

Risk: High

Confidence: Medium

Parameter: userHash

Attack: userHash OR 1=1 --

Evidence:

CWE ID: 89

WASC ID: 19

Source: Active (40018 - SQL Injection)

Description:

SQL injection may be possible.

Other Info:

The page results were successfully manipulated using the boolean conditions [userHash AND 1=1 --] and [userHash OR 1=1 --]

The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison

Data was NOT returned for the original parameter.

Solution:

Do not trust client side input, even if there is client side validation in place.

In general, type check all data on the server side.

If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by ?'

Reference:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

Alert Tags:

Key	Value
OWASP_2021_A03	https://owasp.org/Top10/A03_2021-Injection/
WSTG-v42-INPV-05	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_...
OWASP_2017_A01	https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html

Alertas de nivel medio

Absence of Anti-CSRF Tokens

URL: http://127.0.0.1:8080/WebGoat/login

Risk: Medium

Confidence: Low

Parameter:

Attack:

Evidence: <form action="/WebGoat/login" method="POST" style="width: 200px;">

CWE ID: 352

WASC ID: 9

Source: Passive (10202 - Absence of Anti-CSRF Tokens)

Description:

No Anti-CSRF tokens were found in a HTML submission form.

A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has

Other Info:

No known Anti-CSRF token [anticsrf, CSRFToken, __RequestVerificationToken, csrfmiddlewaretoken, authenticity_token, OWASP_CSRFTOKEN, anoncsrf, csrf_token, _csrf, _csrfSecret, _csrf_magic, CSRF, _token, _csrf_token] was found in the following HTML form: [Form 1: "exampleInputEmail1""exampleInputPassword1"].

Solution:

Phase: Architecture and Design

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Reference:

<http://projects.webappsec.org/Cross-Site-Request-Forgery>
<http://cwe.mitre.org/data/definitions/352.html>

Alert Tags:

Key	Value
OWASP_2021_A01	https://owasp.org/Top10/A01_2021-Broken_Access_Control/
WSTG-v42-SESS-05	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/Session_Management/Test_04:_Session_Management
OWASP_2017_A05	https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.html

Content Security Policy (CSP) Header Not Set

URL: http://127.0.0.1:8080/WebGoat/login

Risk: Medium

Confidence: High

Parameter:

Attack:

Evidence:

CWE ID: 693

WASC ID: 15

Source: Passive (10038 - Content Security Policy (CSP) Header Not Set)

Description:

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets,

Other Info:

Solution:

Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header, to achieve optimal browser support: "Content-Security-Policy" for Chrome 25+, Firefox 23+ and Safari 7+, "X-Content-Security-Policy" for Firefox 4.0+ and Internet Explorer 10+, and "X-WebKit-CSP" for Chrome 14+ and Safari 6+.

Reference:

https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
https://cheatsheetsseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
<http://www.w3.org/TR/CSP/>

Alert Tags:

Key	Value
OWASP_2021_A05	https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
OWASP_2017_A06	https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html

Parameter Tampering						
JRL: http://127.0.0.1:8080/WebGoat/access-control/hidden-menu						
Risk: Medium						
Confidence: Medium						
Parameter: hiddenMenu1						
Attack:						
Evidence: javax.servlet.http.HttpServlet.service(HttpServletRequest.java:517)\n\tat						
CWE ID: 472						
WASC ID: 20						
Source: Active (40008 - Parameter Tampering)						
Description:						
Parameter manipulation caused an error page or Java stack trace to be displayed. This indicated lack of exception handling and potential areas for further exploit.						
Other Info:						
Solution:						
Identify the cause of the error and fix it. Do not trust client side input and enforce a tight check in the server side. Besides, catch the exception properly. Use a generic 500 error page for internal server error.						
Reference:						
Alert Tags:						
<table border="1"> <thead> <tr> <th>Key</th><th>Value</th></tr> </thead> <tbody> <tr> <td>OWASP_2021_A04</td><td>https://owasp.org/Top10/A04_2021-Insecure_Design/</td></tr> <tr> <td>OWASP_2017_A01</td><td>https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html</td></tr> </tbody> </table>	Key	Value	OWASP_2021_A04	https://owasp.org/Top10/A04_2021-Insecure_Design/	OWASP_2017_A01	https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html
Key	Value					
OWASP_2021_A04	https://owasp.org/Top10/A04_2021-Insecure_Design/					
OWASP_2017_A01	https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html					

3.4. Posibles mitigaciones

3.5. Herramientas utilizadas

Las herramientas que se han utilizado para poder realizar la práctica son:

La interfaz gráfica ZAP
Herramientas de Developer