

Introducción a la ciberseguridad

PRÁCTICA CON WEBGOAT

REALIZADA POR

ANGIE ARISTIZABAL BERNAL



KEEPCODING 2022

Sobre el informe

WEBGOAT

Este informe se basa en la aplicación web WebGoat que esta hecha con fines educativos.

Para esto he usado un usuario sin privilegios y estamos hablando de un informe con el concepto de caja gris.

La aplicación web :

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

Las vulnerabilidades **más severas** identificadas durante las pruebas han sido:

- **SQL Injection** : es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos
- **Insecure Direct Object Reference** : la posibilidad de interceptar las peticiones para poder modificar los datos de un usuario.
- **Missing Function Level Access Control** : permite a los usuarios realizar funciones que deberían estar restringidas, o les permite acceder a recursos que deberían estar protegidos.
- **Cross Site Scripting** : puede permitir a una tercera persona inyectar en páginas web visitadas por el usuario código JavaScript o en otro lenguaje similar..

HERRAMIENTAS A USAR

Para realizar estos ejercicios haré uso de las siguientes herramientas:

- **Burp Suite** : es una plataforma para realizar pruebas de seguridad en aplicaciones.

Para su configuración deje todo en modo default, poniendo como objetivo WebGoat y modifique el Proxy de mi navegador poniendo la dirección y puerto de WebGoat

CLASIFICACIÓN DE RIESGOS

Clasificaré las vulnerabilidades en una escala de cinco puntos que muestran la probabilidad de explotación y el riesgo de cada vulnerabilidad.

CRÍTICA – la explotación de esta vulnerabilidad pone en riesgo la confidencialidad y seguridad del sistema. Esta suele ser directa, el atacante no necesita apenas trabajo para realizarla. Estas vulnerabilidades deben ser solucionadas en la mayor brevedad posible para no seguir comprometiendo su seguridad.

ALTA – la explotación de esta vulnerabilidad compromete datos sensibles igual que las clasificadas como críticas, sin embargo, lo que se necesita para realizar estos ataques, ya sean cuentas de usuario y demás, hacen que la dificultad para el atacante para cumplir con su objetivo sea un poco más elevado. Estas vulnerabilidades son igualmente fáciles de explotar pero comprometen menos al servidor.

MEDIA – la explotación de esta vulnerabilidad depende de factores externos ajenos a la seguridad del sistema, como que una persona con acceso a contenido importante sea víctima de phishing u otro tipo de ataque es decir que esta explotación tiene un nivel más alto de dificultad. Los datos que se comprometen con este tipo de vulnerabilidades son menos importantes.

BAJA – la explotación de esta vulnerabilidad compromete de menor forma la aplicación. Es más complicado para el atacante cumplir su objetivo ya que puede depender incluso de acceso físico al servidor para lograr su cometido.

INFO – Estas no son vulnerabilidades como tal. El objetivo de esta etiqueta es señalar puntos que pueden ser útiles o incluso soluciones para implementar la seguridad del sistema y disminuir el nivel de peligro que causan otras vulnerabilidades.

Vulnerabilidades de Web Goat :

SQL Injection - NIVEL : ALTO

A1 - Apartado 10

La aplicación testeada es vulnerable a inyección de código SQL. Haciendo una consulta SQL un atacante es capaz de obtener información de todos los usuarios con unos simples pasos que explicaré a continuación.

¿Que se necesita para explotar esta vulnerabilidad?

- Una cuenta en la página web

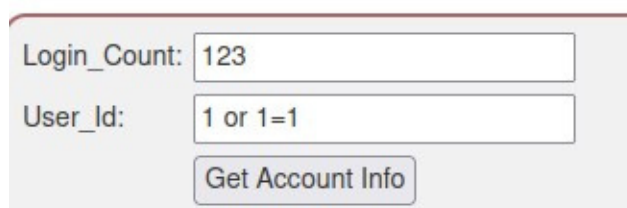
Pasos a seguir:

- Ingresar al siguiente enlace ->

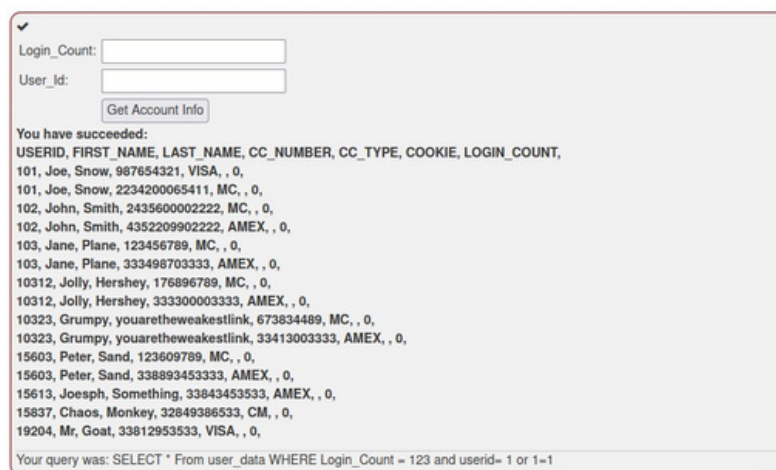
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/9>

o bien irnos al apartado de Injection > SQL Injection (intro) > Punto 10

- Nos encontraremos con dos campos a rellenar : "Login_Count" y "User_Id" . Donde solo nos hará falta escribir cualquier número en el primer campo (Login_Count) e introducir " 1 or 1=1 " en el segundo campo (User_Id) y con esto forzamos a que el resultado sea verdadero cumpliendose siempre la segunda condición como se muestra en la siguiente imagen:



- Una vez rellenados estos campos, sólo hará falta pulsar al botón "Get Account Info" para que se muestra una lista con la información de todos los usuarios :



Recomendación

Validación de entrada y WAF

Escribe un código que pueda identificar a los usuarios ilegítimos. Sin embargo, cuando se utiliza solo, este no es un método infalible.

Puede generar muchos falsos positivos. La implementación de este método junto con el uso de un firewall de aplicaciones web (WAF) puede ser efectivo. El WAF filtrará la inyección SQL y otras amenazas online.

Cuando el WAF detecta un posible usuario ilegítimo, verificará los datos de la IP antes de bloquear la solicitud. Así, se bloquearán los datos de las IPs que tengan mala fama.

En el siguiente enlace podrá encontrar **más información sobre el SQL Injection** y como protegerse de este :

<https://pressroom.hostalia.com/white-papers/ataques-inyeccion-sql/>

SQL Injection – NIVEL : ALTO

A1 – Apartado 11

La aplicación testeada es vulnerable a inyección de código SQL. Haciendo una consulta SQL un atacante es capaz de obtener información de todos los empleados de la empresa.

¿Que se necesita para explotar esta vulnerabilidad?

- Una cuenta en la página web
-

Pasos a seguir:

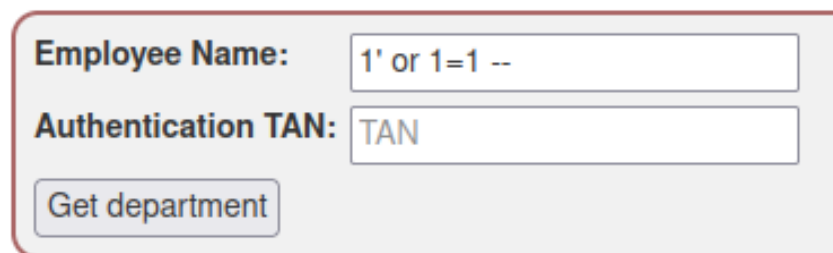
- Ingresar al siguiente enlace ->

<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection.lesson/10>

o bien irnos al apartado de Injection > SQL Injection (intro) > Punto 11

- Nos encontraremos con dos campos a rellenar : "Employee Name" y "Authentication TAN". Donde nuevamente forzamos un True con la siguiente linea de código en el primer campo (Employee Name) : 1' or 1=1 --" .

Como lo muestra la imagen :



Employee Name: 1' or 1=1 --

Authentication TAN: TAN

Get department

- Con esto sólo nos hará falta pulsar "Get department" para que se nos muestre la información de todos los empleados sin si quiera necesitar una contraseña

Recomendación

Validación de entrada y WAF

Escribe un código que pueda identificar a los usuarios ilegítimos. Sin embargo, cuando se utiliza solo, este no es un método infalible.

Puede generar muchos falsos positivos. La implementación de este método junto con el uso de un firewall de aplicaciones web (WAF) puede ser efectivo. El WAF filtrará la inyección SQL y otras amenazas online.

Cuando el WAF detecta un posible usuario ilegítimo, verificará los datos de la IP antes de bloquear la solicitud. Así, se bloquearán los datos de las IPs que tengan mala fama.

En el siguiente enlace podrá encontrar **más información sobre el SQL Injection** y como protegerse de este :

<https://pressroom.hostalia.com/white-papers/ataques-inyeccion-sql/>

Insecure Direct Object References - INFO

A5 - Apartado 3

WebGoat es vulnerable a ataques de referencias directas de objetos inseguras. Un atacante con un proxy como OWASP ZAP puede obtener referencias al funcionamiento interno de la aplicación como el rol y el identificador del usuario.

¿Que se necesita para explotar esta vulnerabilidad?

- Una cuenta en la página web
- Burp Suite u otra aplicación similar

Pasos a seguir:

- Abrimos Burp Suite y lo usamos para usar su navegador y este nos de la información de cada petición que se realice. Con este navegador nos dirigimos al siguiente enlace ->
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/IDOR.lesson/1>
- *Aquí ingresamos con los datos de "Tom" y con la contraseña de "Cat"*



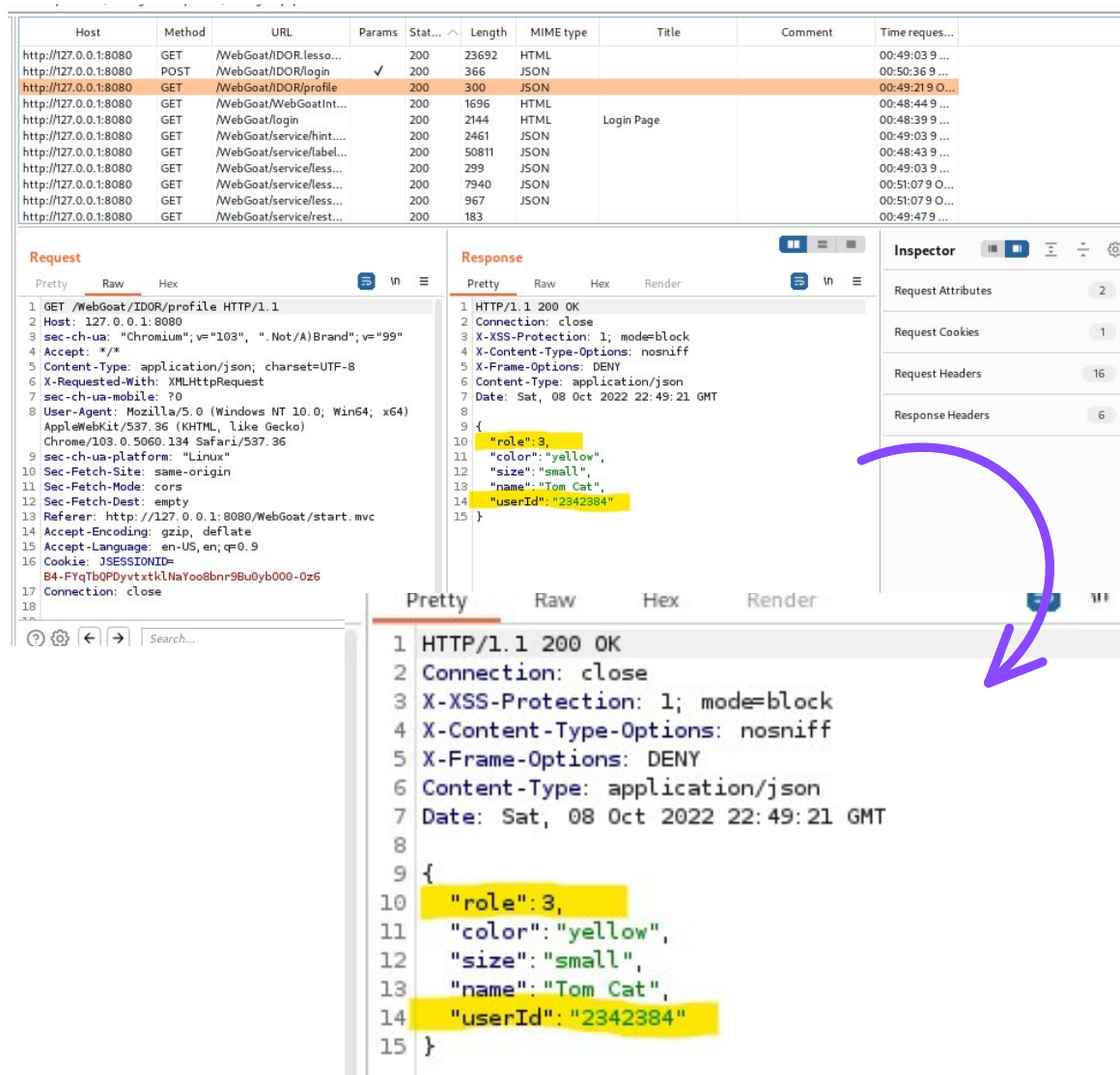
- Una vez dentro, nos vamos al siguiente enlace ->
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/IDOR.lesson/2>
o directamente le damos al botón para ir al ejercicio 3.
- Aquí nos encontramos con dos botones : "View Profile" y "Submit Diffs". Le daremos al primer botón ("View Profile"). Este nos mostrará tres diferentes datos, tales como : nombre, color y talla.



- Con esto hecho, nos iremos al apartado de Target (Site Map) en Burp Suite y buscaremos la petición (GET) al siguiente URL :

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

Una vez hallado, veremos su respectiva respuesta donde veremos que nos proporciona dos datos más aparte del nombre, color y talla. Recibimos "role" y "userId"



The screenshot displays the Burp Suite interface. The top panel shows a list of HTTP requests. The selected request is a GET request to `/WebGoat/IDOR/profile` with a status of 200 and a MIME type of JSON. The bottom panel shows the details of this request and response.

Request Details:

- Method: GET
- URL: `/WebGoat/IDOR/profile`
- Status: 200
- Length: 300
- MIME type: JSON

Response Details:

```

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: Sat, 08 Oct 2022 22:49:21 GMT
8
9 {
10   "role": 3,
11   "color": "yellow",
12   "size": "small",
13   "name": "Tom Cat",
14   "userId": "2342384"
15 }

```

A blue arrow points from the "role" and "userId" fields in the JSON response to the "Inspector" panel on the right, which shows the request attributes, cookies, headers, and response headers.

Recomendación

No mostrar de ninguna forma referencias a información confidencial como contraseñas o nombres de archivo

Si desea más información sobre esta vulnerabilidad y como evitarla, visite la siguiente página:

<https://blog.hackmetrix.com/insecure-direct-object-reference/#:~:text=El%20IDOR%20es%20un%20tipo,el%20debido%20control%20de%20acceso.>

Insecure Direct Object References - INFO

A5 - Apartado 4

WebGoat es vulnerable a ataques de referencias directas de objetos inseguras. Un atacante con un proxy como Burp Suite puede obtener referencias o directamente información sobre información privada del usuario tales como su rol o su número de identificación y con esto acceder a la ruta alternativa al perfil del usuario

¿Que se necesita para explotar esta vulnerabilidad?

- Una cuenta en la página web
- Burp Suite u otra aplicación similar

Pasos a seguir:

- Abrimos Burp Suite y nuevamente abrimos el navegador desde este sin olvidar la información que recogimos en la vulnerabilidad anterior.
- Nos vamos a la siguiente página :
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/IDOR.lesson/3>
o directamente pulsamos al botón correspondiente para ir al apartado 4.
- Aquí nos encontramos con un campo por rellenar. En este pondremos lo siguiente : WebGoat/IDOR/profile/2342384 y así completaremos el url a gracias a haber visto anteriormente el ID del usuario por medio de BurpSuite.

Please input the alternate path to the Url to view your own profile. Please start with 'WebGoat' (i.e. disregard 'http://localhost:8080/')

- Una vez dentro, sólo queda pulsar "Submit" para ver la información del perfil.



Please input the alternate path to the Url to view your own profile. Please start with 'WebGoat' (i.e. disregard 'http://localhost:8080/')

Congratulations, you have used the alternate Url/route to view your own profile.

(role=3, color=yellow, size=small, name=Tom Cat, userId=2342384)

Recomendación

No mostrar de ninguna forma referencias a información confidencial como contraseñas o nombres de archivo

Si desea más información sobre esta vulnerabilidad y como evitarla, visite la siguiente página:

<https://blog.hackmetrix.com/insecure-direct-object-reference/#:~:text=El%20IDOR%20es%20un%20tipo,el%20debido%20control%20de%20acceso.>

Insecure Direct Object References - ALTA

A5 - Apartado 5

WebGoat es vulnerable a ataques de referencias directas de objetos inseguras. Un atacante con un proxy como Burp Suite puede ver los perfiles de otros usuarios y cambiar la información de estos.

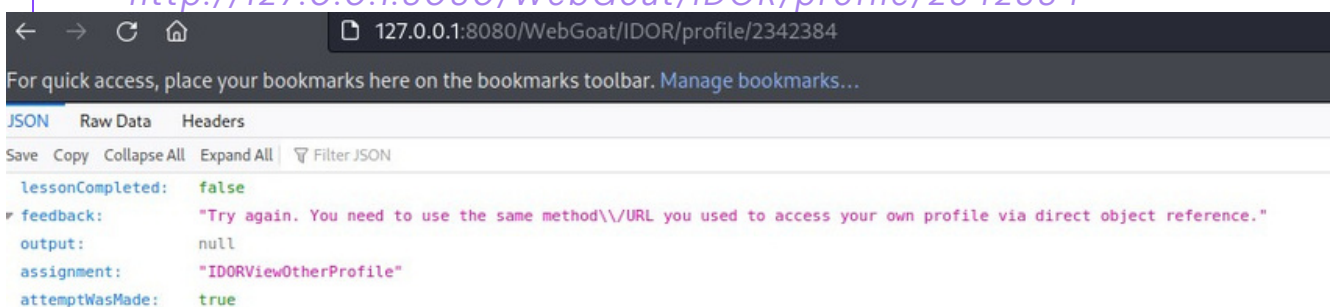
¿Que se necesita para explotar esta vulnerabilidad?

- Una cuenta en la página web
- Burp Suite u otra aplicación similar

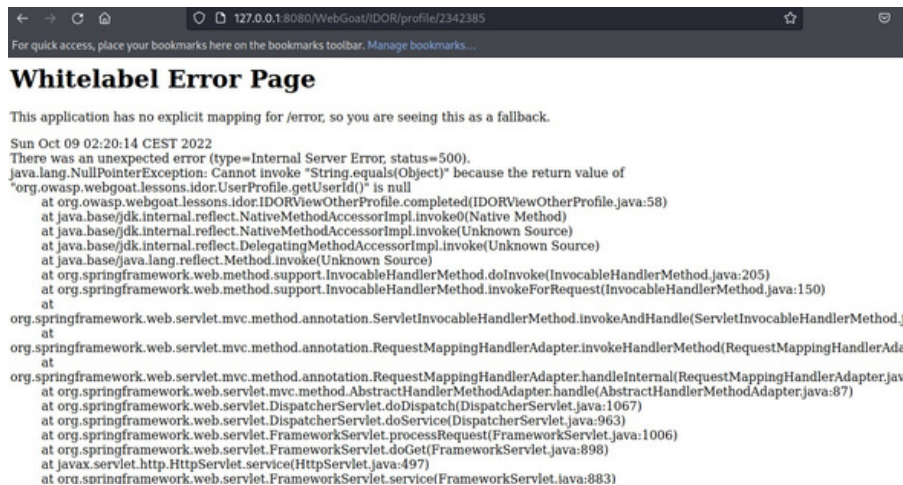
Pasos a seguir:

- Nos vamos a la siguiente página que gracias a la vulnerabilidad anterior sabemos que el url es de la siguiente manera :

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

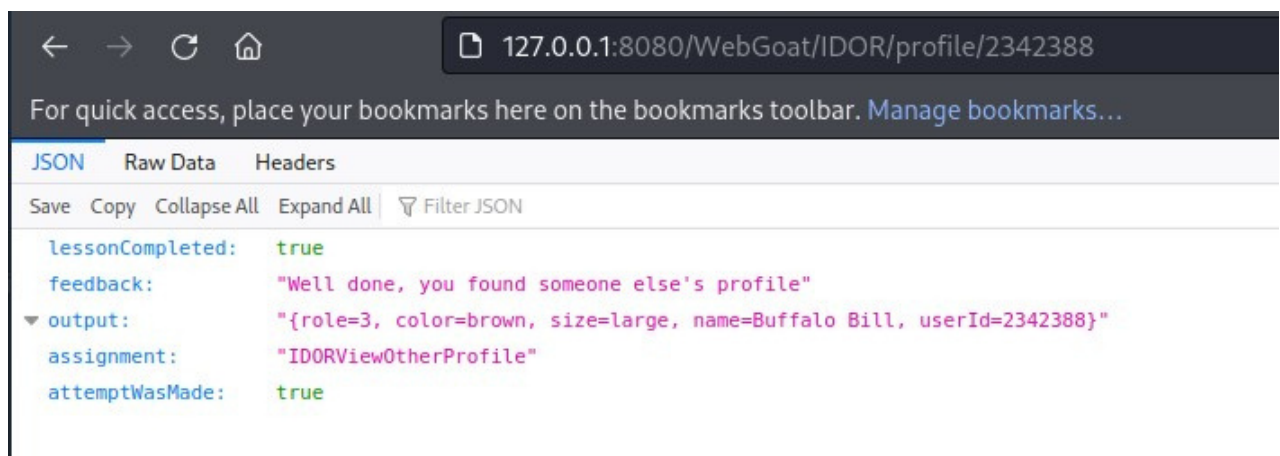


- De aquí puedo partir a hacer fuerza bruta con el mismo buscador, en busca del ID de un usuario



- Hasta dar con la siguiente página :

<http://127.0.0.1:8080/WebGoat/IDOR/profile/2342388>



- Una vez aquí puedo ver el perfil de otro usuario:

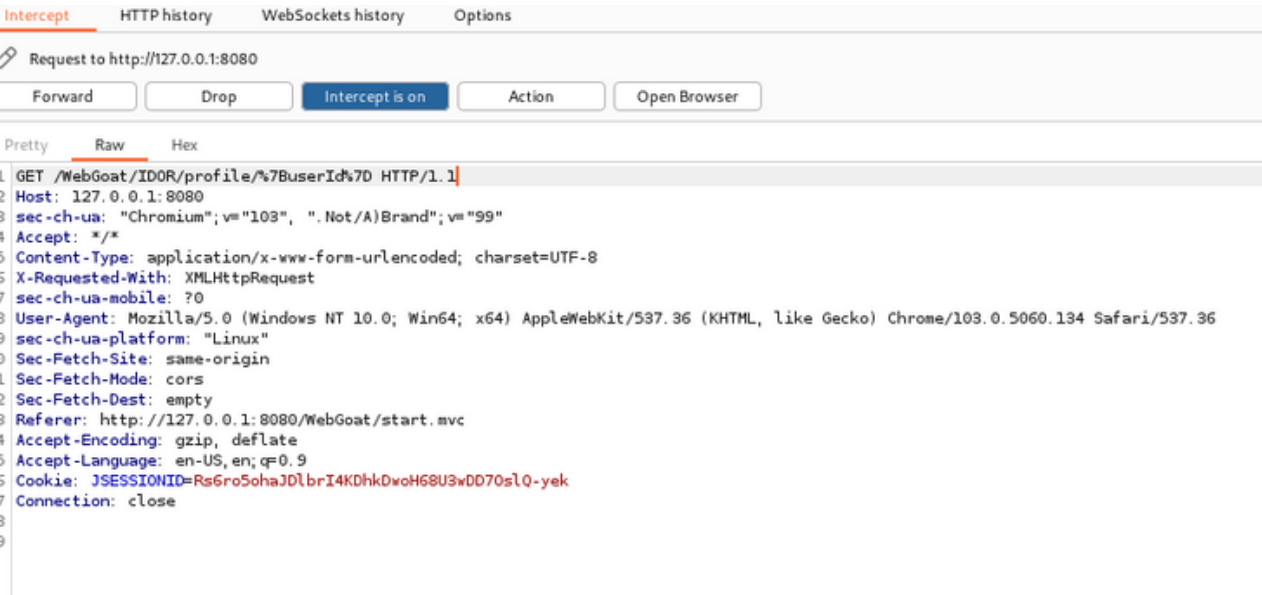
"output" : "{role=3, color=brown, size=large, name=Buffalo Bill, userId=2342388}",
 "assignment" : "IDORViewOtherProfile",
 "attemptWasMade" : true

- <http://127.0.0.1:8080/WebGoat/start.mvc#lesson/IDOR.lesson/4>

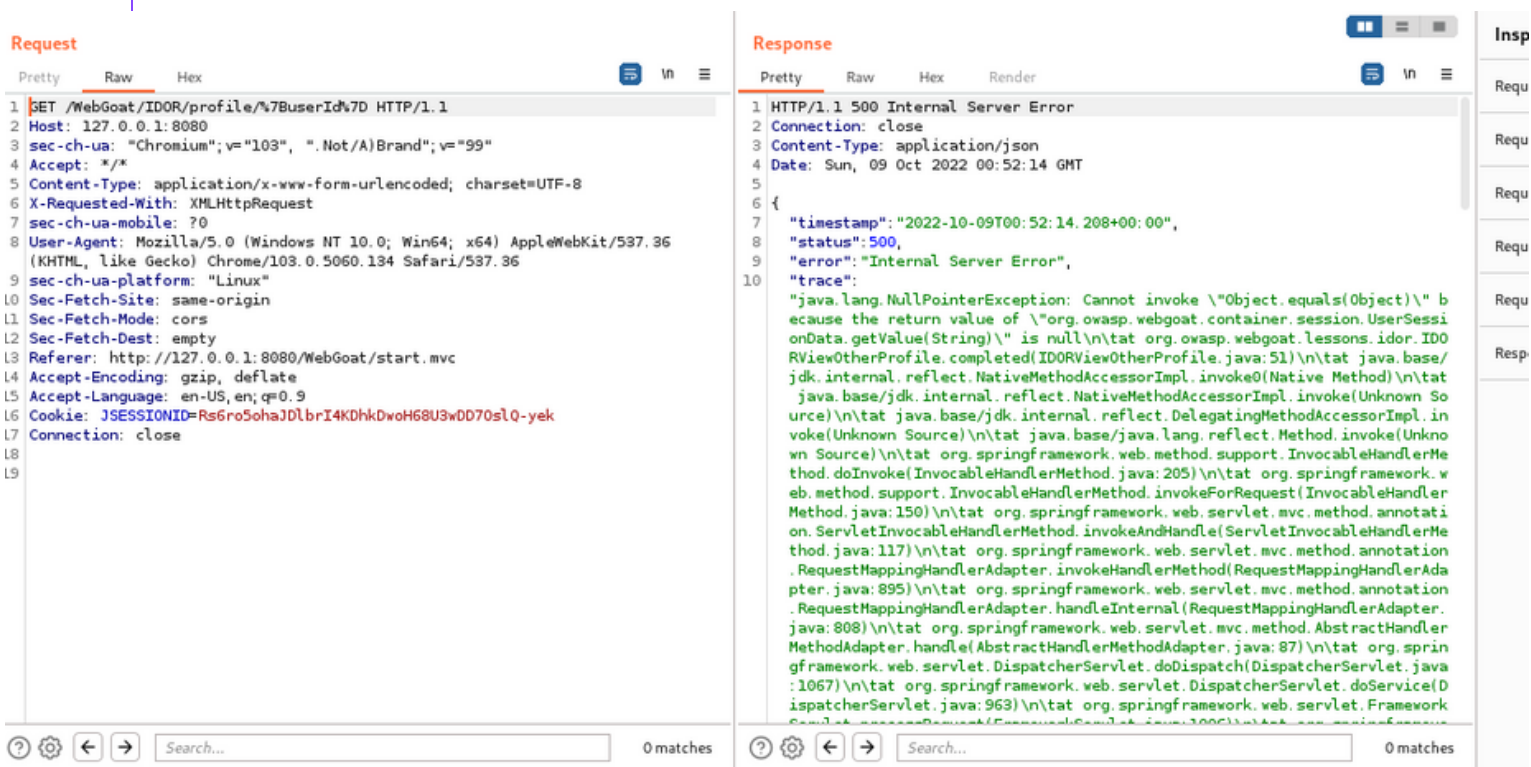
o bien dando click directamente al apartado 5.

- Una vez aquí le damos click al segundo botón de "View Profile", el cual se encuentra justo debajo del apartado "Edit another profile".
- Con esta petición hecha, nos dirigimos a Burp Suite y buscamos la siguiente petición :

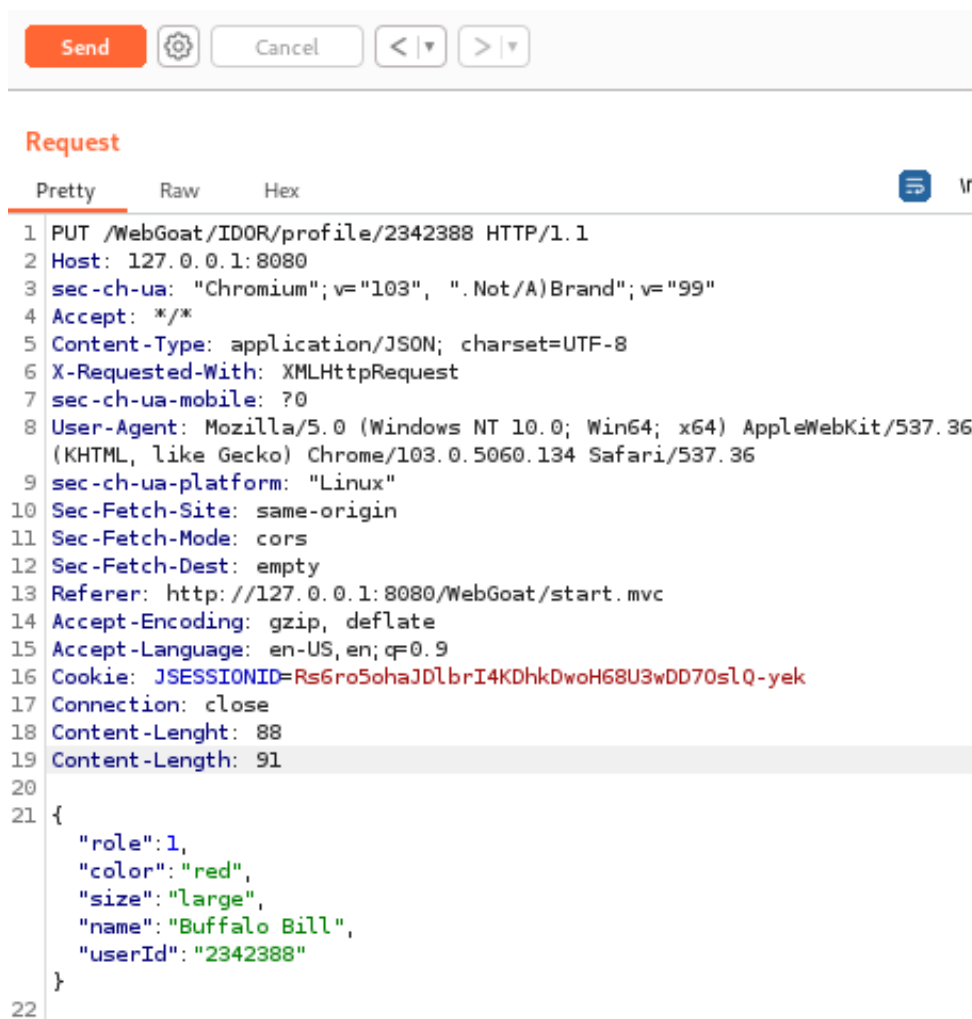
GET /WebGoat/IDOR/profile/%7BuserId%7D HTTP/1.1



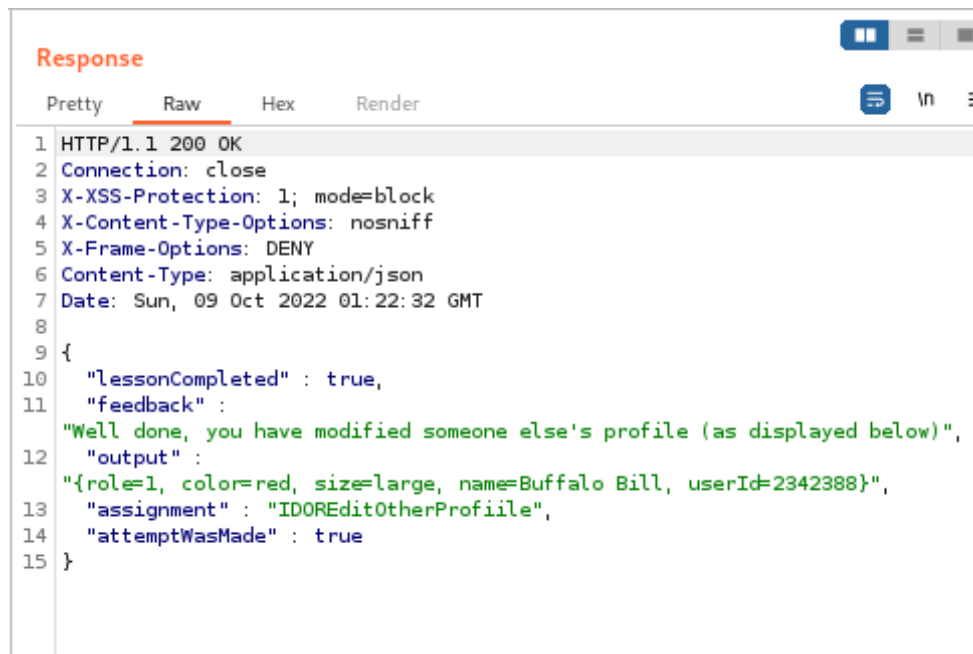
- Y enviamos esto al Repeater



- Cambio el "GET /WebGoat/IDOR/profile/%7BuserId%7D HTTP/1.1" por "PUT /WebGoat/IDOR/profile/2342388 HTTP/1.1"
- Reemplazo "Content-Type: application/x-www-form-urlencoded; charset=UTF-8 " por "Content-Type: application/JSON; charset=UTF-8"
- Añado la siguiente línea :
Content-Length:88
{ "role":1, "color":"red", "size":"large", "name":"Buffalo Bill","userId":2342388 }



- Sólo queda hacer click en "Send" , consiguiendo la siguiente respuesta



```
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

No mostrar de ninguna forma referencias a información confidencial como contraseñas o nombres de archivo.

Reforzar el control de acceso a las consultas y no está de mas utilizar JSON Web Tokens para verificar la integridad de estos.

Si desea más información sobre esta vulnerabilidad y como evitarla, visite la siguiente página:

<https://blog.hackmetrix.com/insecure-direct-object-reference/#:~:text=El%20IDOR%20es%20un%20tipo,el%20debido%20control%20de%20acceso.>

Missing Function Level Access Control - INFO

A5 - Apartado 2

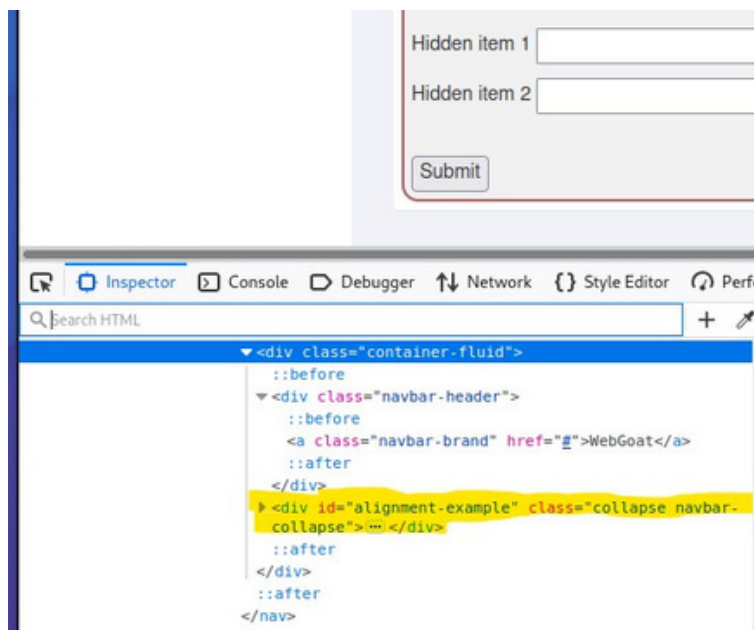
WebGoat es vulnerable a ataques de falta de control de acceso a nivel de función. Cualquier persona con intenciones maliciosas podría ver dentro del código de la página y encontrar enlaces que estan "ocultos" y no deberían poderlos ver nadie que no este autorizado.

¿Que se necesita para explotar esta vulnerabilidad?

- Una cuenta en la página web

Pasos a seguir:

- Nos vamos a la siguiente página :
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/MissingFunctionAC.lesson/1> o directamente pulsamos al botón correspondiente para ir al apartado 2 de Missing Function Level Access Control
- Dentro de este espacio debajo del apartado de "Your mission", más específicamente en el menu daremos click derecho, yendonos a "Inspeccionar".



- Desplegamos el elemento seleccionado
- Desplegamos nuevamente " <ulclass="nav navbar-nav">
- Aquí encontraremos "<li class="hidden-menu-item dropdown"> donde ya nos informa que hay un elemento escondido dentro del menu.

```

    </ul>
  </li>
  <li class="dropdown">... </li>
  <li class="hidden-menu-item dropdown">
    <a class="dropdown-toggle" href="#" data-
      toggle="dropdown" role="button" aria-

```

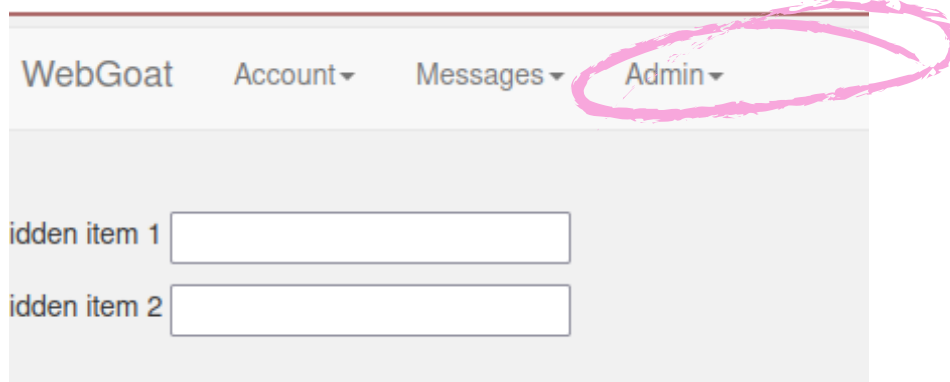
- Aquí cambiamos "class="hidden-menu-item dropdown" por "class="dropdown"

```

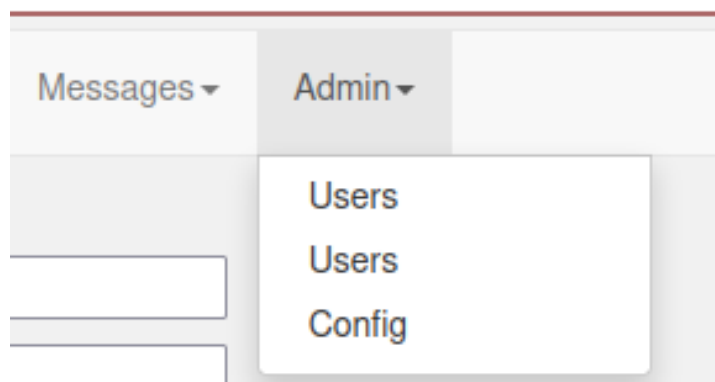
    <a class="dropdown-toggle" href="#" data-toggle="dropdown"
      role="button" aria-haspopup="true" aria-expanded="false">... </a>
    <ul class="dropdown-menu" aria-labelledby="messages">... </ul>
  </li>
  <li class="dropdown">
    <a class="dropdown-toggle" href="#" data-toggle="dropdown"
      role="button"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

```

- Sólo nos queda hacer click en la tecla "Enter" de nuestro teclado y se nos será visible automáticamente el apartado "Admin" que en un principio estaba oculto



- Donde si damos click ya podremos ver de igual manera el sub menu que ocultaba



Recomendación

Elementos que pueden comprometer la seguridad del sistema, tales como un menu de administrador sólo debería ser visto en código HTML por personas autorizadas.

Se podría verificar que usuario es el que esta online para saber si tiene acceso al código HTML o no.

Missing Function Level Access Control - ALTA

A5 - Apartado 3

WebGoat es vulnerable a ataques de falta de control de acceso a nivel de función. Cualquier persona con intenciones maliciosas conociendo las rutas podría obtener los datos de los usuarios comprometiendo la integridad y confidencialidad del sistema.

¿Que se necesita para explotar esta vulnerabilidad?

- Una cuenta en la página web
- BurpSuite o cualquier aplicación similar

Pasos a seguir:

- Nos vamos a la siguiente página desde Burp Suite:
<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/MissingFunctionAC.lesson/2>
o directamente pulsamos al botón correspondiente para ir al apartado 3 de Missing Function Level Access Control
- Una vez aquí le daremos click a "Submit" interceptando esta petición.
- Nos vamos a BurpSuite donde nos mostrará un registro con una petición POST con un 200 OK

http://127.0.0.1:8080	Method	URL	Status	Size	Content-Type	Page
http://127.0.0.1:8080	GET	/WebGoat/MissingFun...	200	8654	HTML	
http://127.0.0.1:8080	GET	/WebGoat/WebGoatInt...	200	1696	HTML	
http://127.0.0.1:8080	POST	/WebGoat/access-cont...	200	390	JSON	
http://127.0.0.1:8080	GET	/WebGoat/login	200	2144	HTML	Login Page
http://127.0.0.1:8080	GET	/WebGoat/service/hint...	200	1669	JSON	
http://127.0.0.1:8080	GET	/WebGoat/service/label...	200	50811	JSON	
http://127.0.0.1:8080	GET	/WebGoat/service/less...	200	326	JSON	
http://127.0.0.1:8080	GET	/WebGoat/service/less...	200	7939	JSON	
http://127.0.0.1:8080	GET	/WebGoat/service/less...	200	710	JSON	

Request

```

1 POST /WebGoat/access-control/user-hash 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/x-www-form-urlencoded; 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=

```

Response

```

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 09:49:40 GMT
8
9 {
10   "lessonCompleted":false,
11   "feedback":
12     "Sorry the solution is not correct, please try again."
13   "output":null,
14   "assignment": "MissingFunctionACYourHash",
15   "attemptWasMade":true
16 }

```

- Esto lo enviaremos al "Repetaer" para editarlo
- Una vez aquí cambiaremos la peticion "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".

Request

```

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=

```

- Sólo nos queda hacer click en Send y recibiremos los nombre de los usuarios con sus respectivos hashes . También podremos ver si estos son administradores o no.

```
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 10:09:23 GMT
8
9 [
10   {
11     "username": "Tom",
12     "admin": false,
13     "userHash": "Mydnhcy00j2b0m6SjmPz6PUxF9WIE07tzm665GiZWCo="
14   },
15   {
16     "username": "Jerry",
17     "admin": true,
18     "userHash": "SVt0Laa+ER+w2eoIIVE5/77umvhcsh5V8UyDLUa1Itg="
19   },
20   {
21     "username": "Sylvester",
22     "admin": false,
23     "userHash": "B5zhk70ZfZLuvQ4smRL4nqCvd0TggMZtKS3TtTqIed0="
24   }
25 ]
```

Recomendación

Elementos que pueden comprometer la seguridad del sistema, tales como un menu de administrador sólo debería ser visto en código HTML por personas autorizadas.

Se podría verificar que usuario es el que esta online para saber si tiene acceso al código HTML o no.

Si desea más información sobre esta vulnerabilidad y como evitarla, visite la siguiente página:

<https://crashtest-security.com/missing-function-level-access-control/>

Cross Site Scripting - ALTA

A7 - Apartado 7

La aplicación testeada es vulnerable a ataques de secuencia de comandos en sitios cruzados. Cualquier persona con intenciones maliciosas puede inyectar código malicioso en la aplicación web.

¿Que se necesita para explotar esta vulnerabilidad?

- Una cuenta en la página web

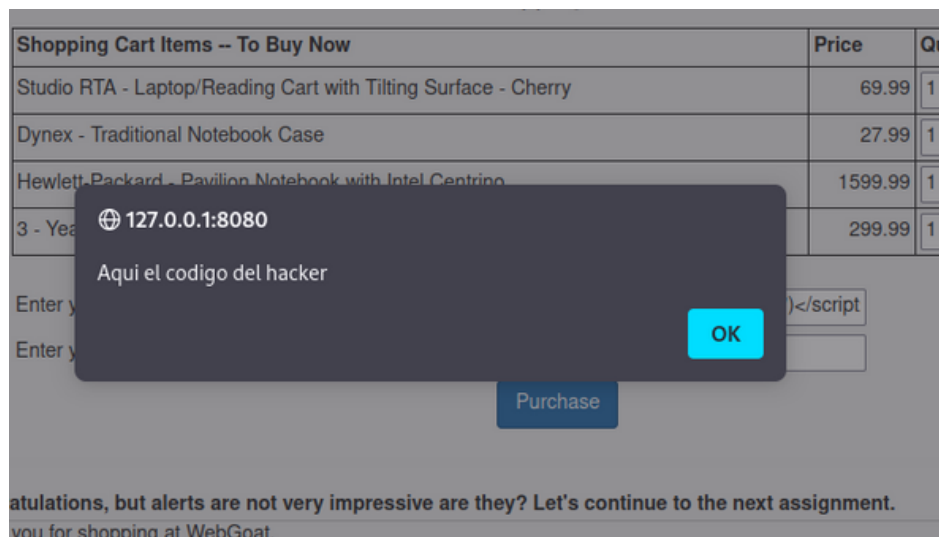
Pasos a seguir:

- Nos vamos a la siguiente página:

<http://127.0.0.1:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/6>

directamente pulsamos al botón correspondiente para ir al apartado 3 de Cross Site Scripting que se encuentra dentro del apartado de "Injection"

- Una vez aquí escribo código JavaScript entre "<script>" y creo una alerta. En mi caso escribo :
`<script>alert('Aquí el código del hacker')</script>`
- Click en "Purchase" y recibo un mensaje emergente que si fuera de un atacante podría ser peligroso



Recomendación

Sanitizar la entrada de datos evitará que un atacante ingrese código con fines maliciosos y cause estragos en el sistema

Si desea más información sobre esta vulnerabilidad y como evitarla, visite la siguiente página:

<https://es.godaddy.com/blog/que-es-el-cross-site-scripting-xss-y-como-puedes-evitarlo/>

INFORMATION GATHERING

Puertos abiertos según Nmap

- 8080/tcp – http-proxy
- 9090/tcp – zeus-admin?

Tecnologías según Wappalyzer :

- Lenguajes de programación:
 - Java
- Frameworks de JS:
 - AlertifyJS
 - RequireJS v2.3.6
- Librerías de JS:
 - core-js v2.4.0
 - DataTables v1.10.11
- Gráficos de JS:
 - Highcharts v5.0.14