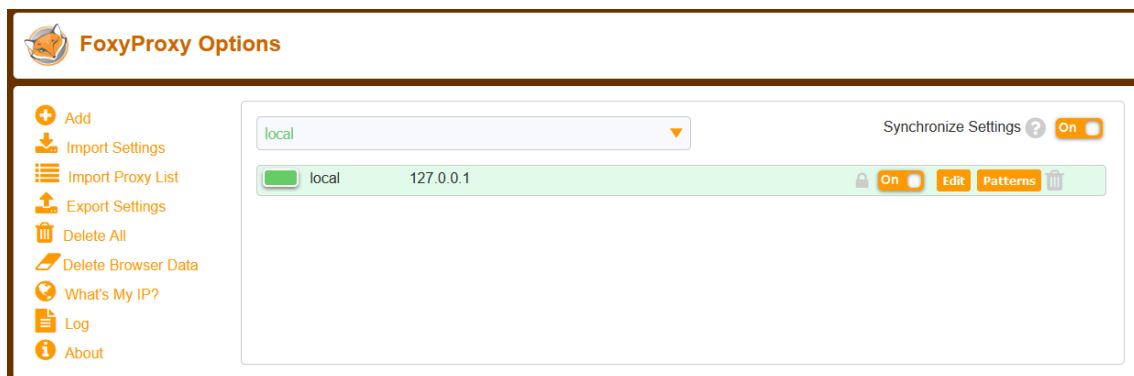


1 A1 Access Broken Control

1.1 Hijack sesión

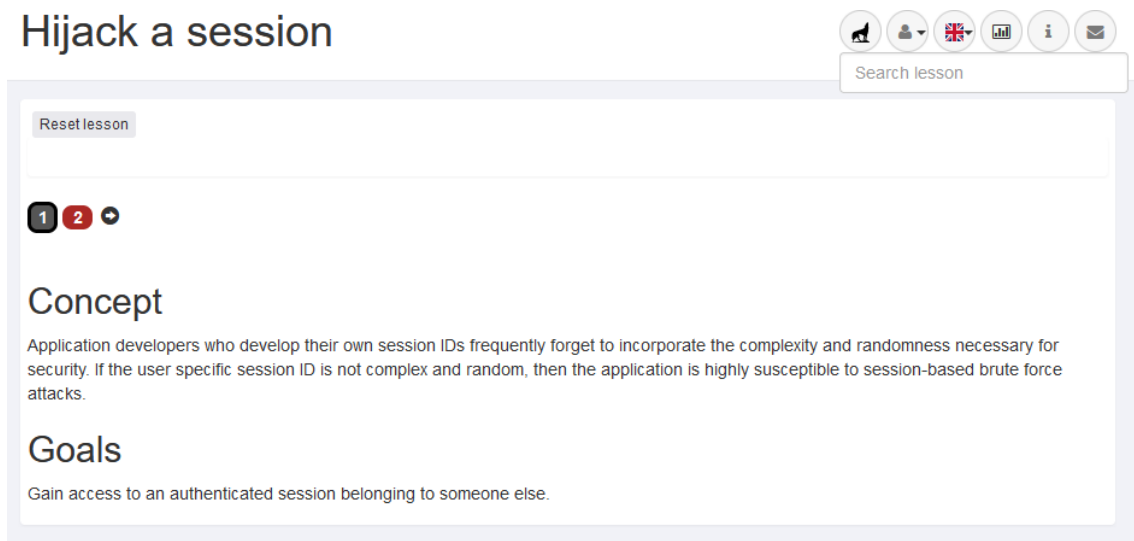
Los desarrolladores de aplicaciones que desarrollan sus propios ID de sesión con frecuencia olvidan incorporar la complejidad y aleatoriedad necesarias para la seguridad. Si el ID de sesión específico del usuario no es complejo ni aleatorio, entonces la aplicación es muy susceptible a ataques de fuerza bruta basados en sesiones

Asegúrate que FoxyProxy este activo en el navegador FireFox.

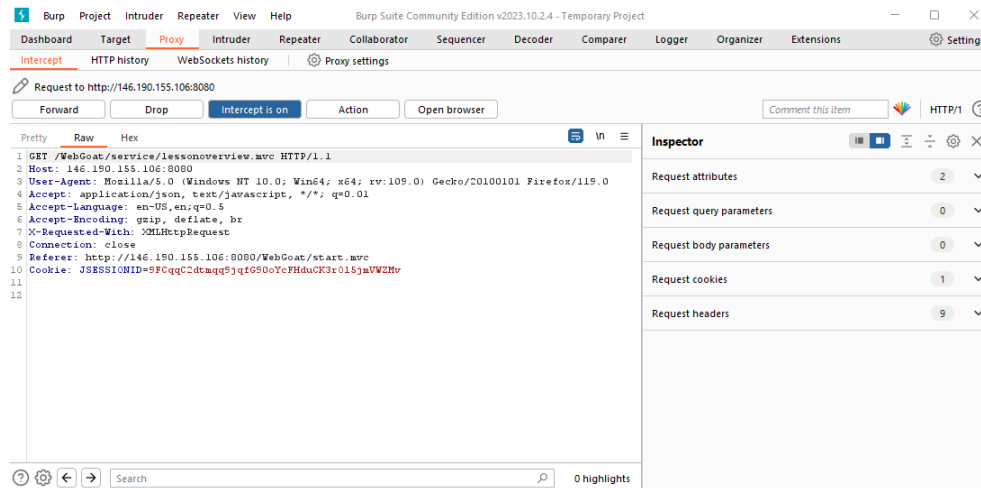


Captura el tráfico con el proxy Burp Suite.

Hijack a session

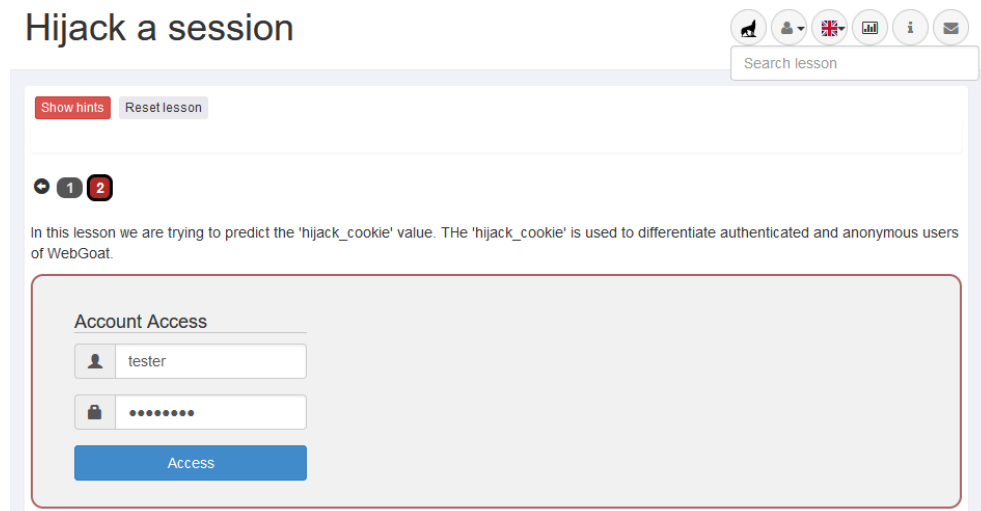


Abre Burp Suite.

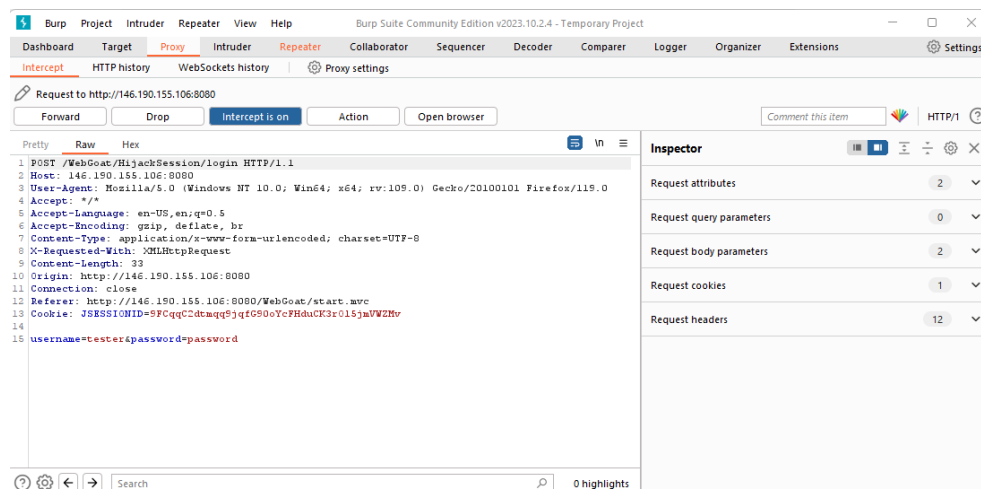


Intenta el login con el usuario actual (tester en este ejemplo).

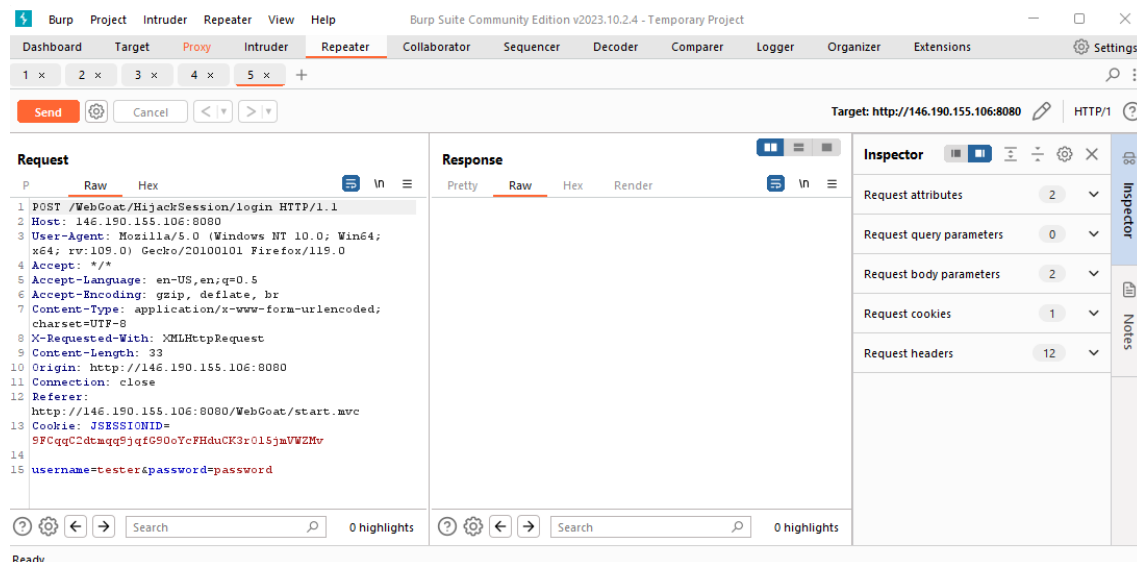
Hijack a session



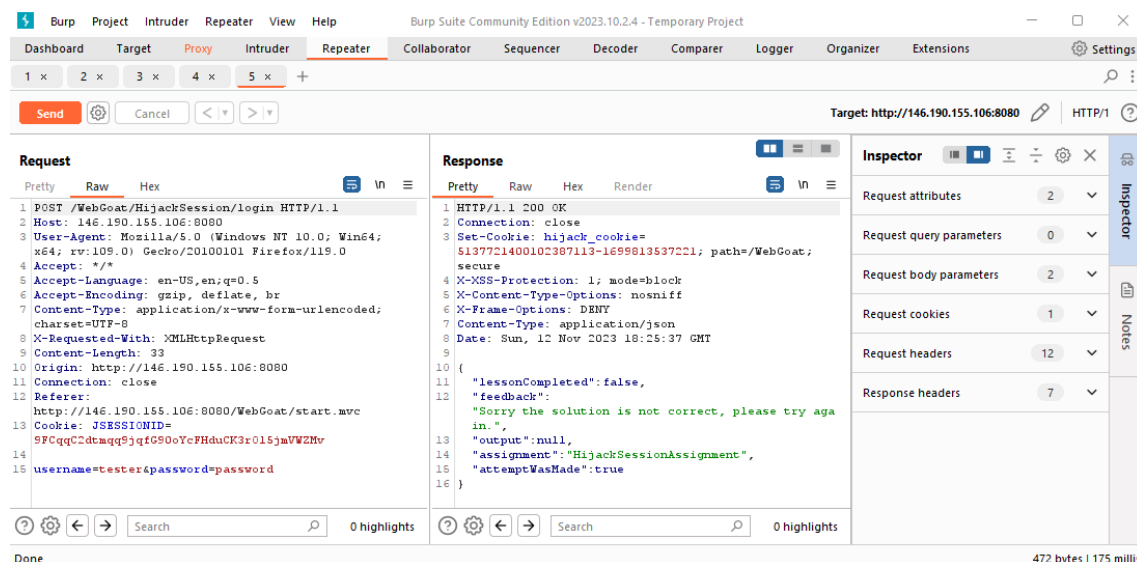
Captura en Burp Suite la petición: “**POST /WebGoat/HijackSession/login HTTP/1.1**”



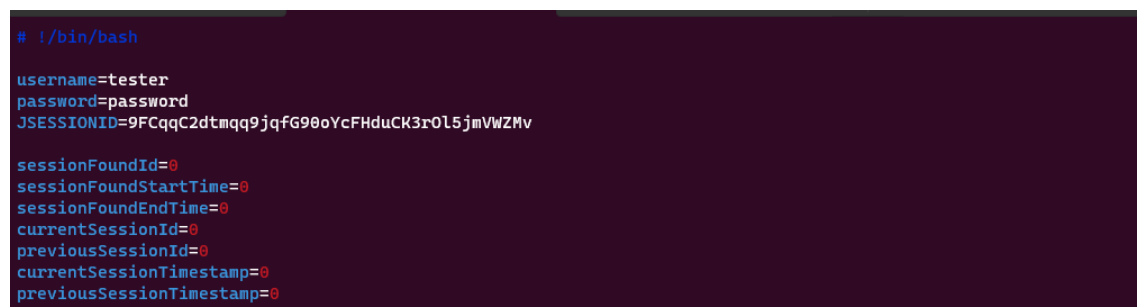
Envía a “**Repeater**” a petición capturada.



Haz Clic en “Send” para obtener el Cookie.



Edita el script get_cookies.sh. agrega el JSESSIONID que obtuviste con BurpSuite.



Ejecuta el script y verifica que retorne el mensaje de éxito.

```
[webgoat@server1 Scripts]$ sh get_cookies.sh
===== Searching for session =====

-
-
-
-
-

Session found: 5137721400102387130 - 5137721400102387132

===== Session Found: 5137721400102387131 =====

| From timestamps 1699815106059 to 1699815106347 |

===== Starting session for 5137721400102387131 at 1699815106059 =====

5137721400102387131-1699815106059: "Congratulations. You have successfully completed the assignment.",
5137721400102387131-1699815106060: "Sorry the solution is not correct, please try again.",
5137721400102387131-1699815106061: "Sorry the solution is not correct, please try again.",
5137721400102387131-1699815106062: "Sorry the solution is not correct, please try again.",
5137721400102387131-1699815106063: "Sorry the solution is not correct, please try again.",
5137721400102387131-1699815106064: "Sorry the solution is not correct, please try again.",
5137721400102387131-1699815106065: "Sorry the solution is not correct, please try again.",
5137721400102387131-1699815106066: "Sorry the solution is not correct, please try again.",
5137721400102387131-1699815106067: "Sorry the solution is not correct, please try again.",
```

1.2 Insecure Direct Object References

Las referencias directas a objetos se producen cuando una aplicación utiliza entradas proporcionadas por el cliente para acceder a datos y objetos.

Ejemplos

Los ejemplos de referencias directas a objetos que utilizan el método GET pueden parecerse a

<https://some.company.tld/dor?id=12345>

<https://some.company.tld/images?img=12345>

<https://some.company.tld/dor/12345>

Estos se consideran inseguros cuando la referencia no se maneja adecuadamente y permite eludir autorizaciones o revelar datos privados que podrían usarse para realizar operaciones o acceder a datos que el usuario no debería poder realizar o acceder. Digamos que, como usuario, vas a ver tu perfil y la URL se parece a:

<https://some.company.tld/app/user/23398>

... y podrás ver tu perfil allí. ¿Qué sucede si navegas a:

<https://some.company.tld/app/user/23399>...o use otro número al final. Si puede manipular el número (id de usuario) y ver el perfil de otra persona, entonces la referencia del objeto es insegura. Por supuesto, esto se puede comprobar o ampliar más allá de los métodos GET para ver datos, pero también para manipularlos.

1.2.1 Observing Differences & Behaviors

Haz login con el usuario indicado.



Authenticate First, Abuse Authorization Later

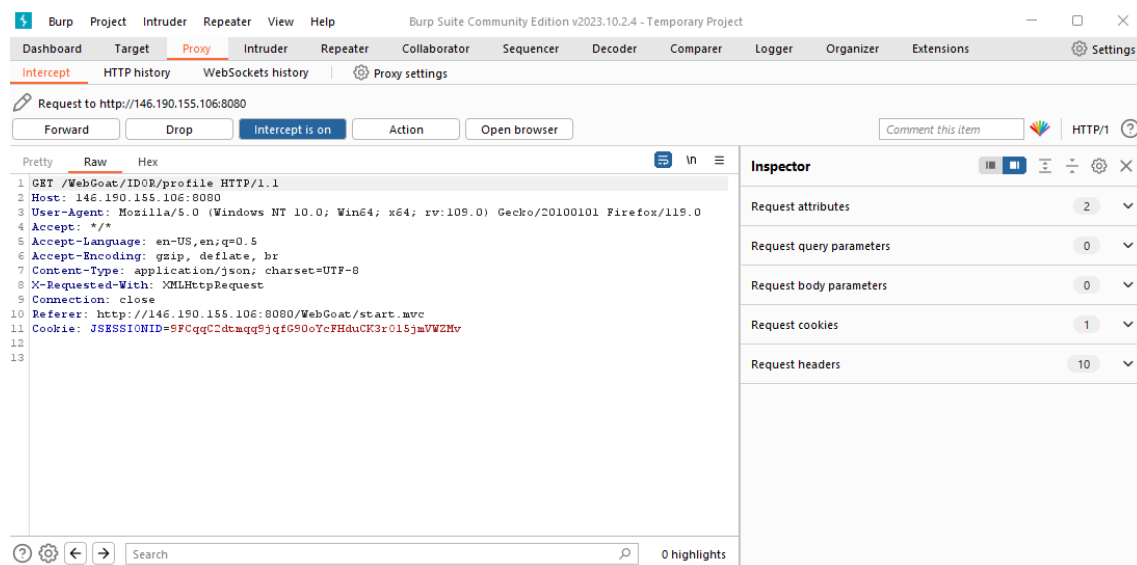
Many access control issues are susceptible to attack from an authenticated-but-unauthorized user. So, let's start by legitimately authenticating. Then, we will look for ways to bypass or abuse Authorization.

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

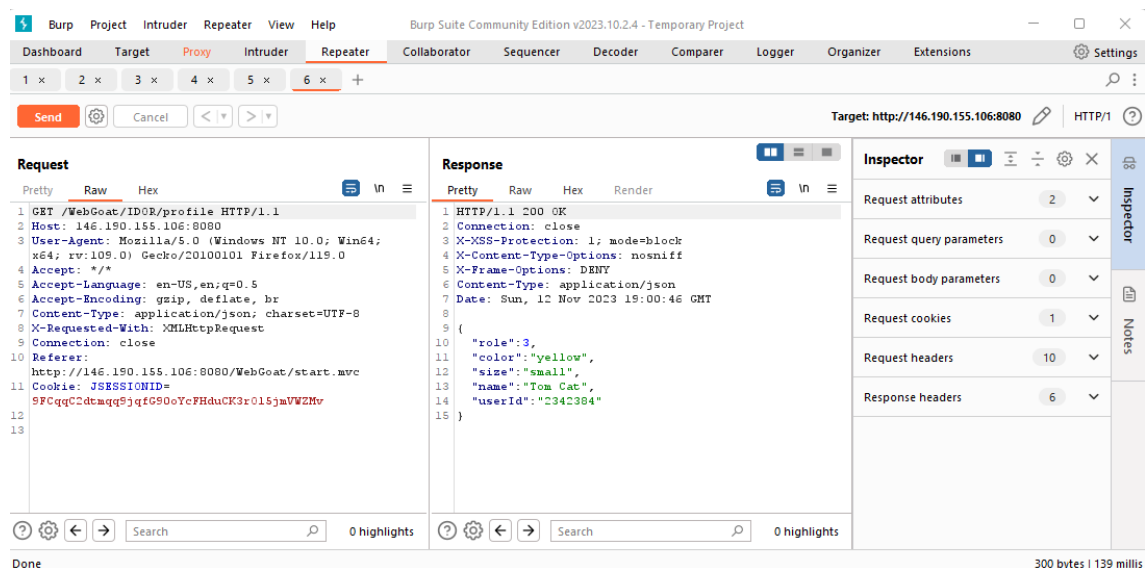
After authenticating, proceed to the next screen.

user/pass user: pass:

A partir de la lección 3, captura el tráfico con el proxy.



Agrega la captura al “Repeater”. En Repeater haz clic en “Send”.



Los campos ocultos que no se muestran en la pagina son: role y userId.

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

Envia tu respuesta. Haz clic en “Submit Diffs”.

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

1.2.2 Guessing & Predicting Patterns

La aplicación con la que estamos trabajando parece seguir un patrón RESTful en lo que respecta al perfil. Muchas aplicaciones tienen funciones en las que un usuario elevado puede acceder al contenido de otro. En ese caso, simplemente /profile no funcionará ya que los datos de sesión/autenticación del propio usuario no nos dirán qué perfil quieren ver. Entonces, ¿cuál crees que es un patrón probable para ver tu propio perfil explícitamente usando una referencia de objeto directa?

Analizando el “Repeater” del ejercicio anterior.

The screenshot shows the Burp Suite Repeater interface. The Request tab is active, displaying an HTTP GET request to /WebGoat/IDOR/profile. The Response tab shows a 200 OK response with a JSON body. The Inspector panel on the right highlights the 'role' attribute in the JSON body, which has a value of 3. The Request attributes panel also shows the 'role' attribute with a value of 3, along with other attributes like 'color', 'size', 'name', and 'userId'.

OWASP

Observamos que el patron es:

/WebGoat/IDOR/profile/2342384

1 2 3 4 5 6

Guessing & Predicting Patterns

View Your Own Profile Another Way

The application we are working with seems to follow a RESTful pattern so far as the profile goes. Many apps have roles in which an elevated user may access content of another. In that case, just /profile won't work since the own user's session/authentication data won't tell us whose profile they want view. So, what do you think is a likely pattern to view your own profile explicitly using a direct object reference?

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

WebGoat/IDOR/profile/234; Submit

Haz Clic en "Submit".

✓

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

WebGoat/ Submit

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

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

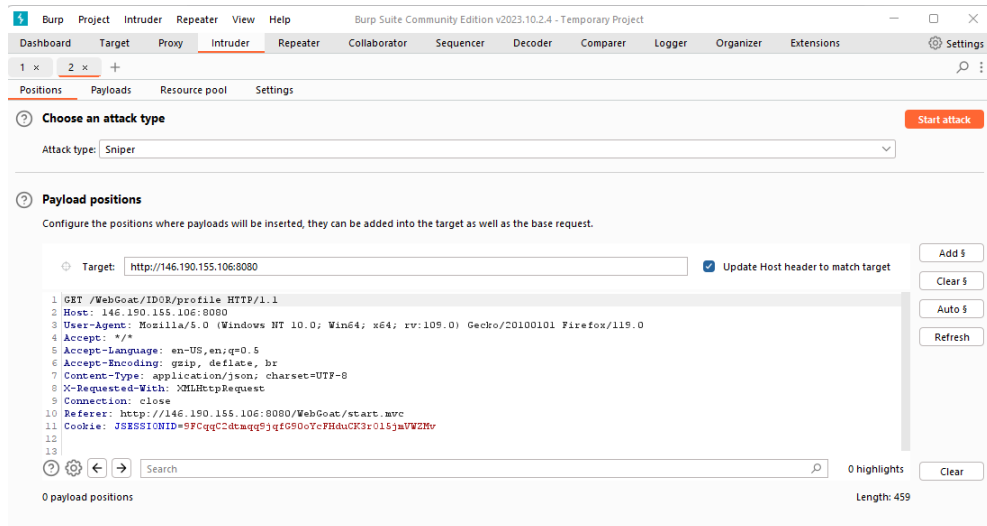
1.2.3 Playing with the Patterns

Vea el perfil de otra persona utilizando la ruta alternativa que ya utilizó para ver su propio perfil. Utilice el botón 'Ver perfil' e intercepte/modifique la solicitud para ver otro perfil. Alternativamente, es posible que también pueda utilizar una solicitud GET manual con su navegador.

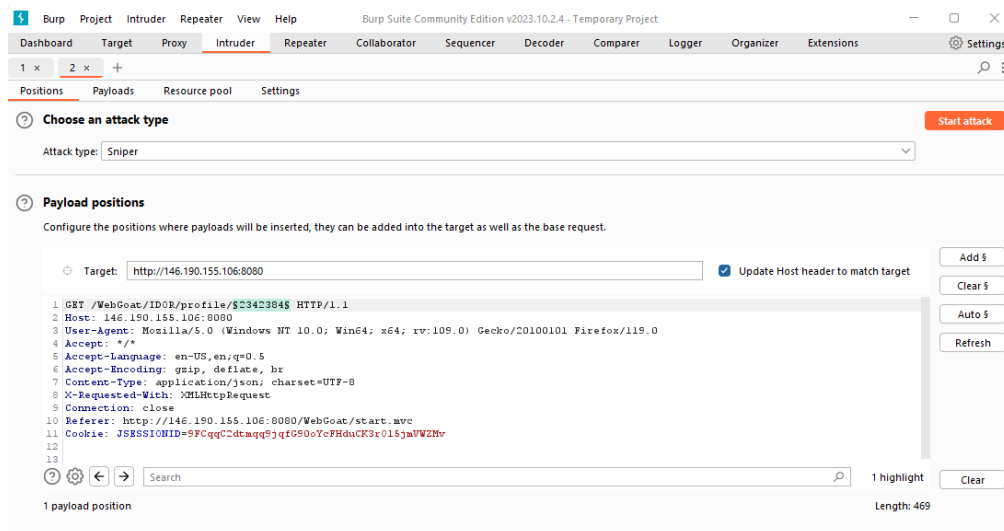
Nuestro perfil lo podemos ver:

The screenshot shows the Burp Suite interface with a GET request intercepted. The request is to `http://146.190.155.106:8080/WebGoat/IDOR/profile`. The response is a JSON object: `{ "role": 3, "color": "yellow", "size": "small", "name": "Tom Cat", "userId": "2342384" }`. The right-hand pane shows the 'Inspector' tab with request and response details.

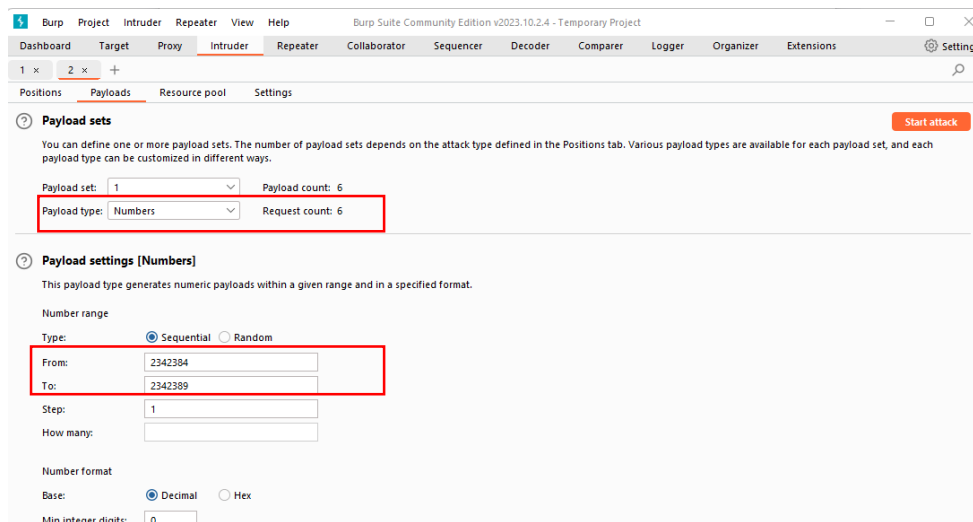
¿Pero podemos ver el perfil de otro usuario?, no deberías poder verlo pero un ataque de intrusión lo podría conseguir. Agrega la petición de "Repeater" a "Intruder".



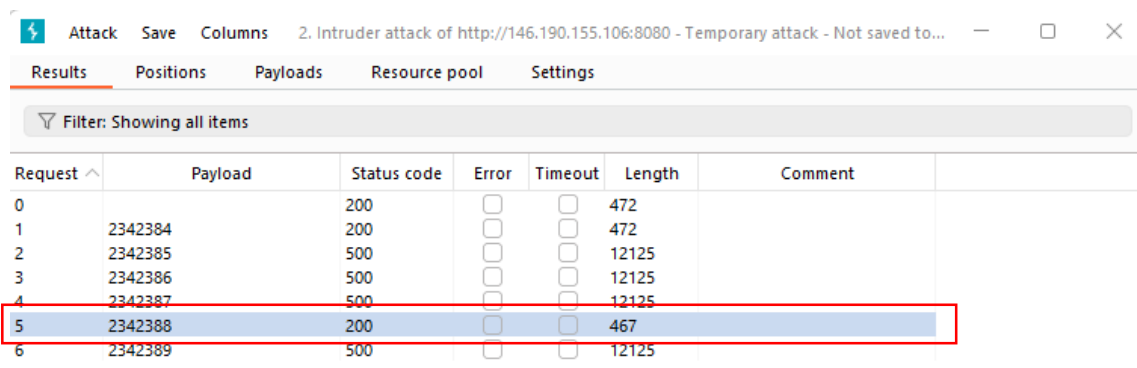
Agrega tu userId en "Intruder".



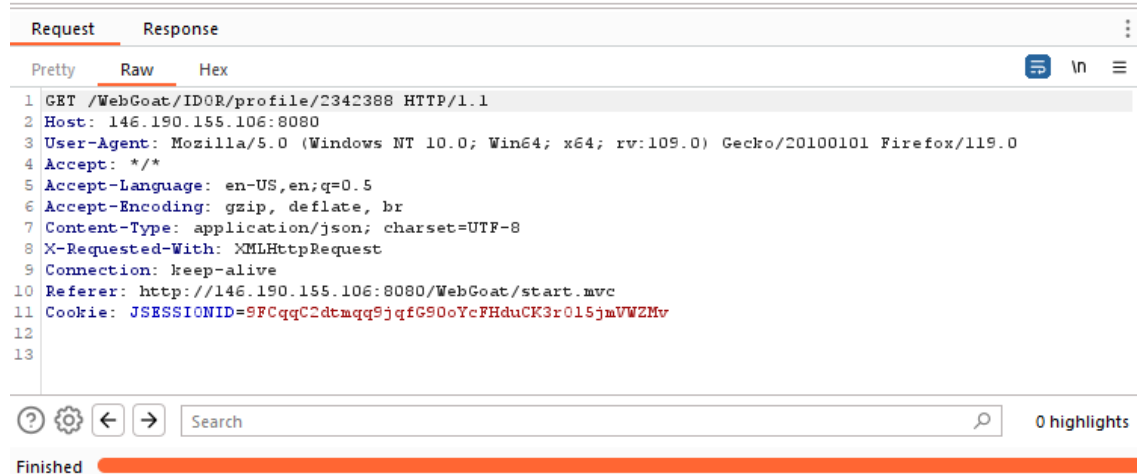
En Intruder>>Payloads. En Payload type asigna **Numbers**



Haz clic en “**Star Attack**” y encuentra un nuevo usuario.



Request	Payload	Status code	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	472	
1	2342384	200	<input type="checkbox"/>	<input type="checkbox"/>	472	
2	2342385	500	<input type="checkbox"/>	<input type="checkbox"/>	12125	
3	2342386	500	<input type="checkbox"/>	<input type="checkbox"/>	12125	
4	2342387	500	<input type="checkbox"/>	<input type="checkbox"/>	12125	
5	2342388	200	<input type="checkbox"/>	<input type="checkbox"/>	467	
6	2342389	500	<input type="checkbox"/>	<input type="checkbox"/>	12125	



Request Response

Pretty Raw Hex

```

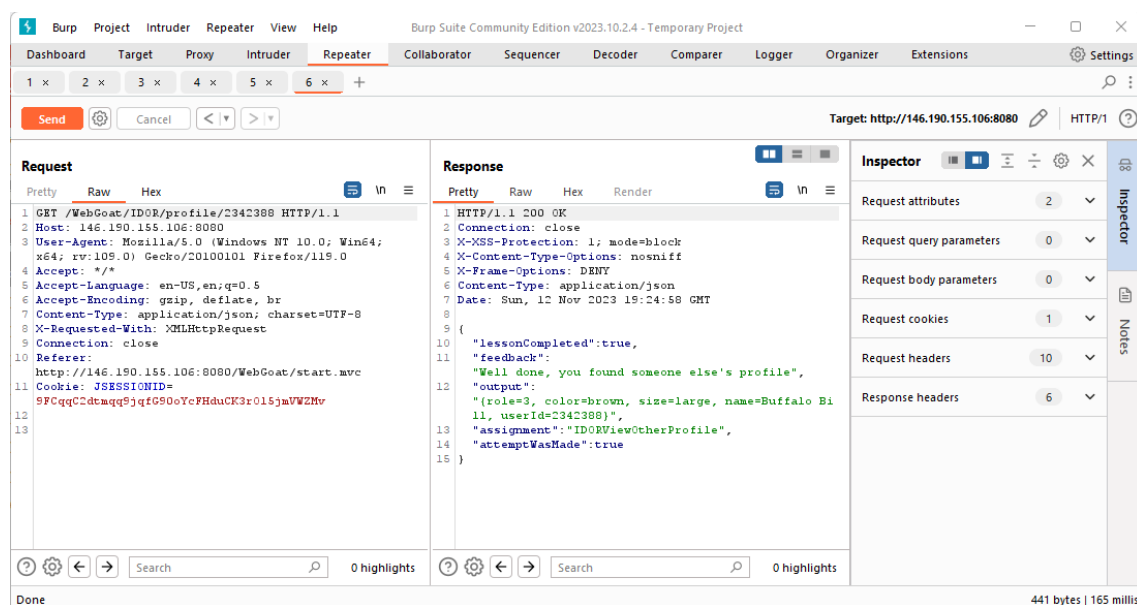
1 GET /WebGoat/IDOR/profile/2342388 HTTP/1.1
2 Host: 146.190.155.106:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Connection: keep-alive
10 Referer: http://146.190.155.106:8080/WebGoat/start.mvc
11 Cookie: JSESSIONID=9FCqqC2dtmqg9jqfG90oYcFHduCK3r015jmVWZMv
12
13

```

Search 0 highlights

Finished

Eureka!, encontramos un usuario con código 234288. Usa “Repeater” para obtener los datos del perfil de este usuario.



Burp Suite Community Edition v2023.10.2.4 - Temporary Project

Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Settings

1 x 2 x 3 x 4 x 5 x 6 x +

Send Cancel < >

Target: http://146.190.155.106:8080 HTTP/1

Request Response

Pretty Raw Hex

```

1 GET /WebGoat/IDOR/profile/2342388 HTTP/1.1
2 Host: 146.190.155.106:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Connection: close
10 Referer: http://146.190.155.106:8080/WebGoat/start.mvc
11 Cookie: JSESSIONID=9FCqqC2dtmqg9jqfG90oYcFHduCK3r015jmVWZMv
12
13

```

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, 12 Nov 2023 19:24:58 GMT
8
9 {
10   "lessonCompleted":true,
11   "feedback":
12     "Well done, you found someone else's profile",
13   "output":
14     "({role=3, color=brown, size=large, name=Buffalo Bi
15     ll, userId=2342388})",
16   "assignment": "IDORViewOtherProfile",
17   "attemptWasMade":true
18 }
19

```

Inspector

Request attributes 2

Request query parameters 0

Request body parameters 0

Request cookies 1

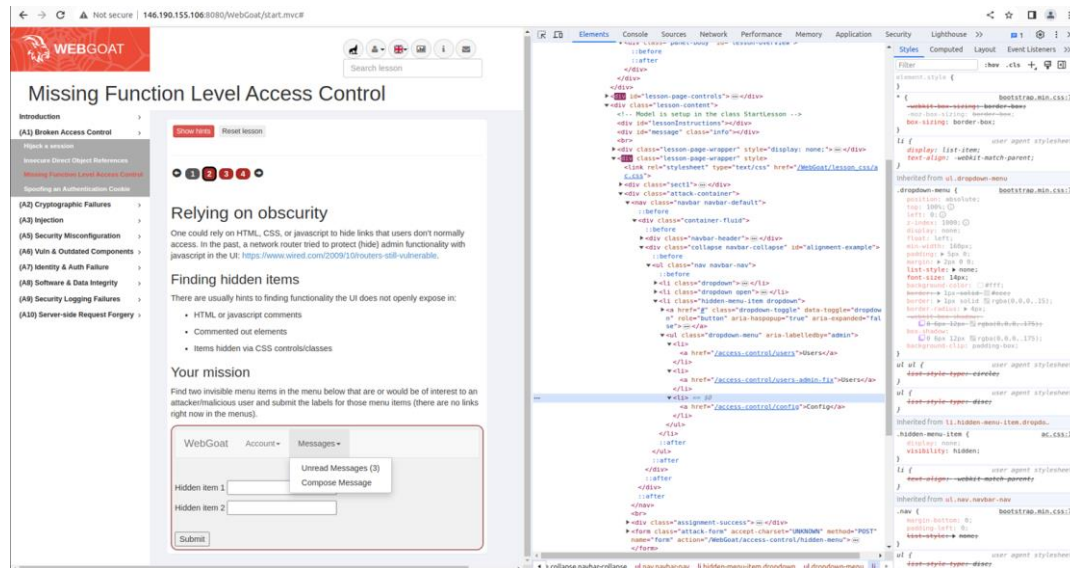
Request headers 10

Response headers 6

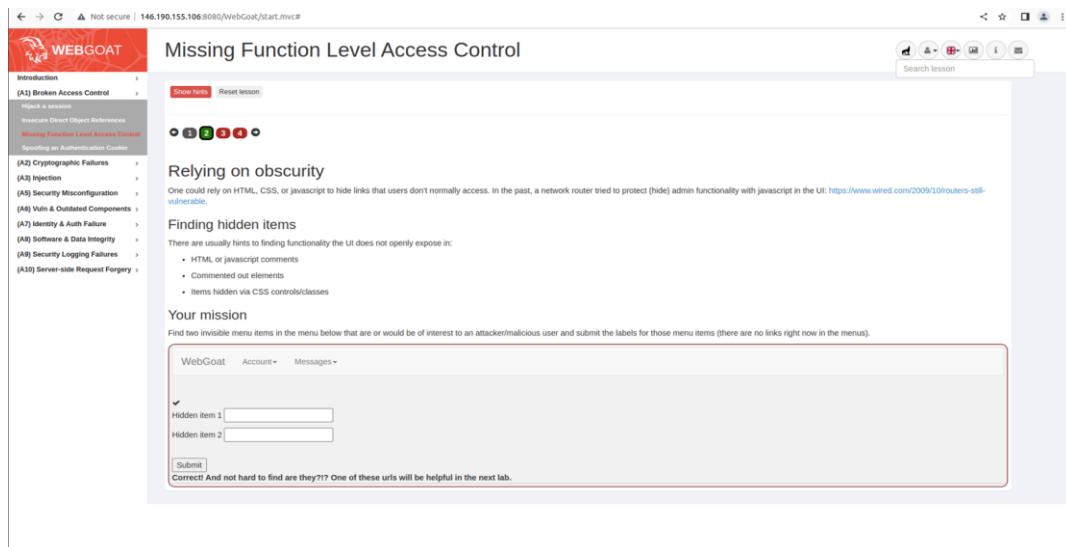
Done 441 bytes | 165 millis

1.3 Missing Function Level Access Control

Los parámetros ocultos son **Users** y **Config**



Verifica introduciendo los parámetros en el formulario.



Encontrar el hash del usuario.

Burp Suite Community Edition v2023.10.2.5 - Temporary Project

Target: <http://146.190.155.106:8080> HTTP/1.1

Request

```

1 GET /WebGoat/access-control/users HTTP/1.1
2 Host: 146.190.155.106:8080
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0)
  Gecko/20100101 Firefox/110.0
4 Content-Type: application/json;charset=UTF-8
5 Accept: application/json, text/javascript, */*; q=0.01
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 X-Requested-With: XMLHttpRequest
9 Connection: close
10 Referer: http://146.190.155.106:8080/WebGoat/start.mvc
11 Cookie: JSESSIONID=uyaSPG-dfsh0s57KAgVzumBdJkBSenPRrZXPhmV
12
13

```

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: Wed, 08 Nov 2023 15:19:30 GMT
8
9 [
10   {
11     "username": "Tom",
12     "admin": false,
13     "userHash": "Mydnhcy00j2b0m65j#Pz6FUXF5W1e07tzm65GjZWCo="
14   },
15   {
16     "username": "Jerry",
17     "admin": true,
18     "userHash": "5vt0Laa+ER+w2e0IVES/77umvhcsh5V8UyQLUa1Itgs="
19   },
20   {
21     "username": "Sylvester",
22     "admin": false,
23     "userHash": "B5zhk70ZfZLuvQ4smRl4nqCvdOTggMZtK53TtTq1ed0="
24   }
25 ]

```

Inspector: Request attributes (2), Request query parameters (0), Request body parameters (0), Request cookies (1), Request headers (10), Response headers (6). 529 bytes | 125 mills.

Copia y pega el hash del usuario **Jerry**.

WEBGOAT

Missing Function Le

Try it

As the previous page described, sometimes applications rely on client-side controls to control access (obscurely). If you can find invisible items, try them and see what happens. Yes, it can be that simple!

Gathering User Info

Often data dumps originate from vulnerabilities such as SQL injection, but they can also come from poor or lacking access control.

It will likely take multiple steps and multiple attempts to get this one:

- Pay attention to the comments and leaked info.
- You may need to use another browser/account along the way.

Start with the information you already gathered (hidden menu item) to see if you can pull the list of users and then provide the hash for Jerry's account.

Your Hash:

Submit

Congrats! You really succeeded when you added the user.

Elements: `<div class="collapse navbar-collapse" id="alignment-example">...</div>`

Styles: `.hidden-menu-item { display: none; visibility: hidden; }`

1.4 Spoofing an Authentication Cookie

Las cookies de autenticación se utilizan para servicios que requieren autenticación, cuando el usuario inicia sesión con un nombre de usuario y contraseña personales, el servidor valida las credenciales proporcionadas y, si son válidas, crea una sesión.

Cada sesión suele tener un ID único que identifica la sesión del usuario; cuando el servidor devuelve la respuesta al usuario, incluye un encabezado Set-Cookie que contiene, entre otras cosas, el nombre y el valor de la cookie.

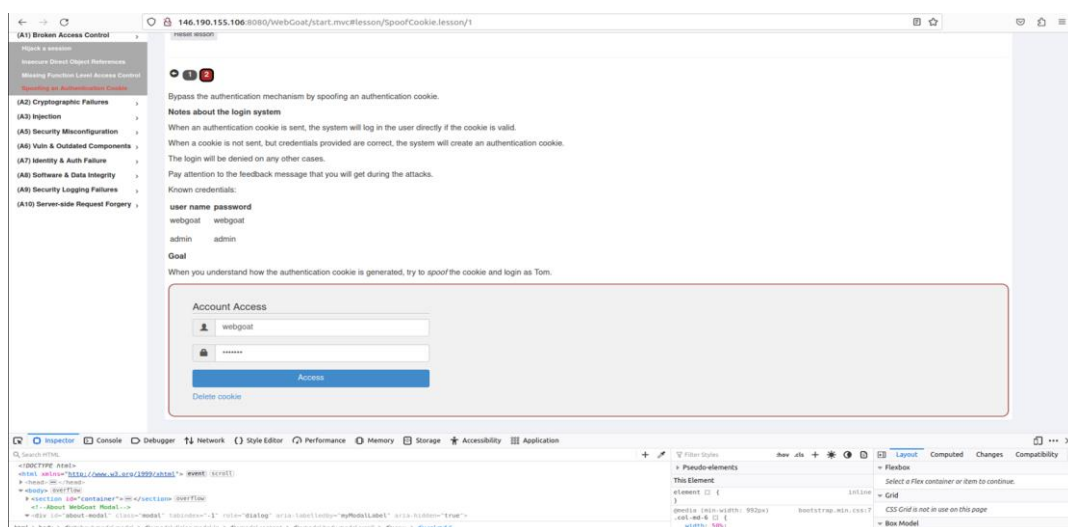
La cookie de autenticación normalmente se almacena en el lado del cliente y del servidor.

Por un lado, tener la cookie almacenada del lado del cliente implica que puede ser robada aprovechando determinadas vulnerabilidades o interceptada mediante ataques man in the middle o XSS. Por otro lado, los valores de las cookies se pueden adivinar si se puede obtener el algoritmo para generarlas.

Muchas aplicaciones iniciarán automáticamente la sesión de un usuario si se proporciona la cookie de autenticación correcta.

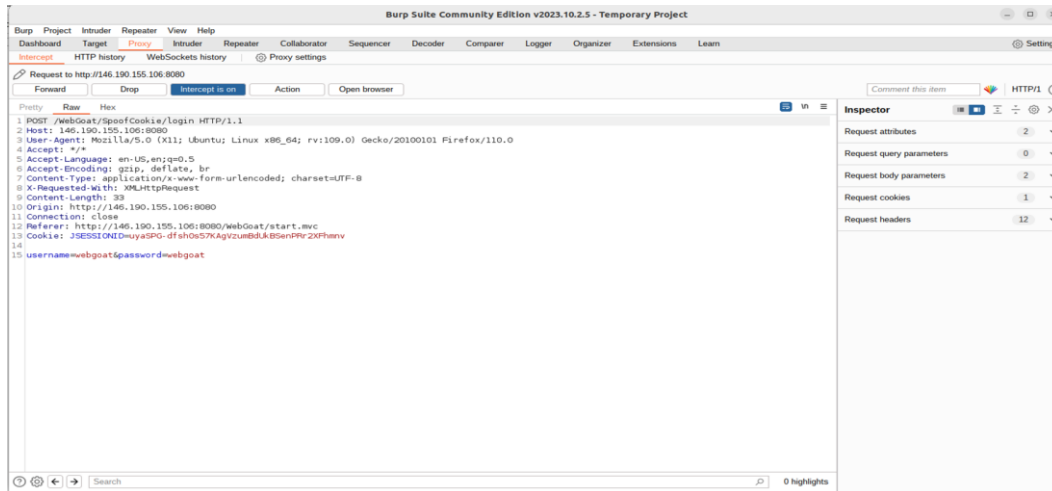
El usuario debería poder adivinar el algoritmo de generación de cookies y evitar el mecanismo de autenticación iniciando sesión como un usuario diferente.

Ingresa con el usuario webgoat clave webgoat, captura la petición con Burp suite.

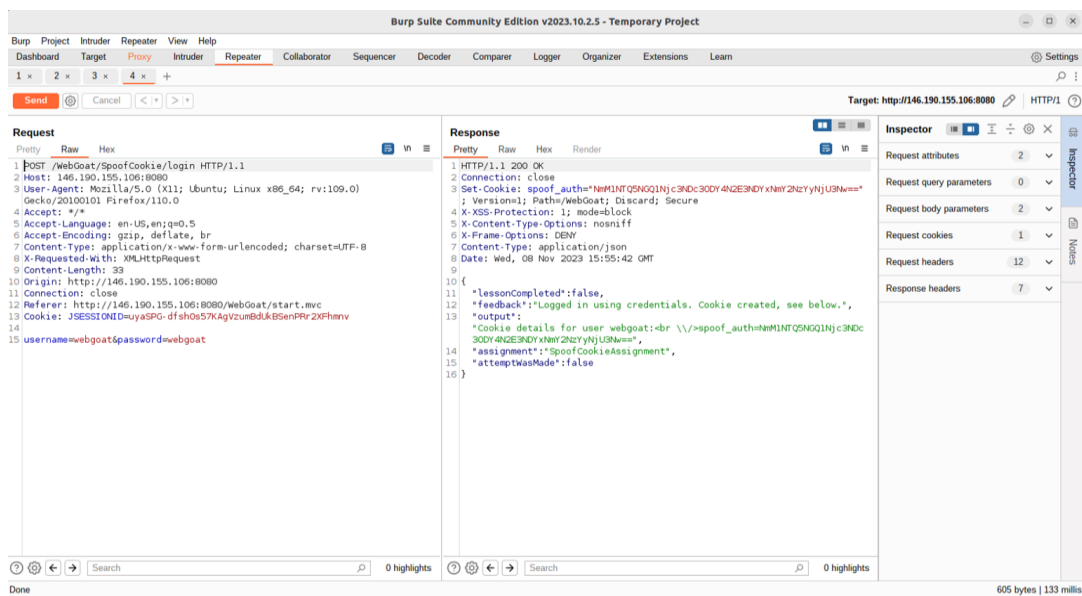


OWASP

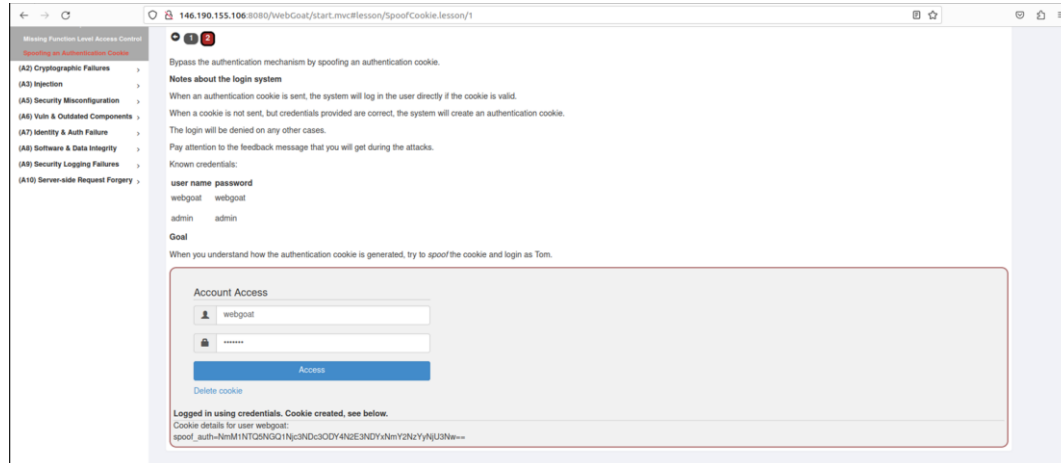
Verifica la petición capturada en Burp suite.



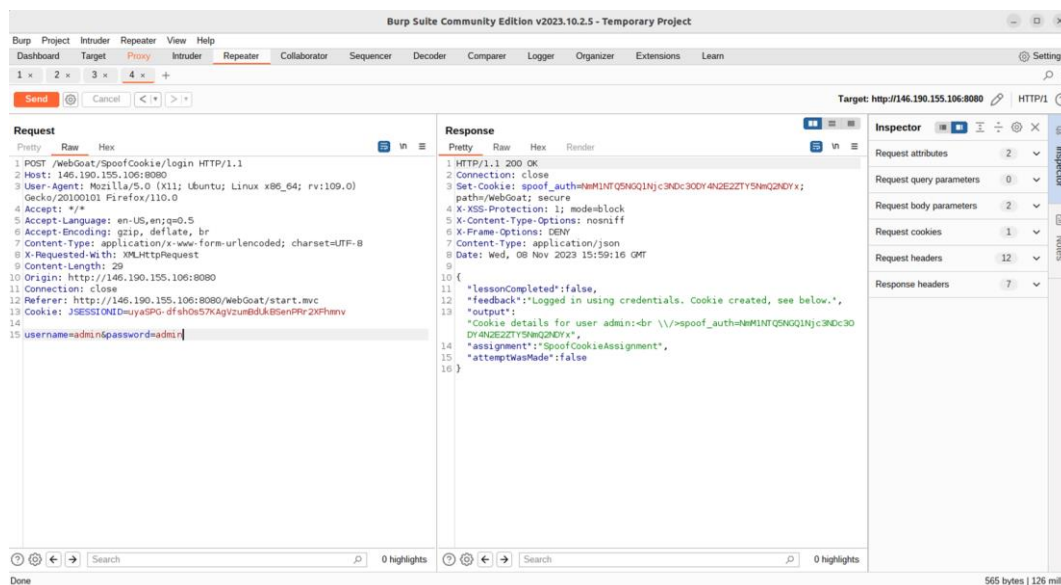
Agrega al “Repeater” la petición capturada.



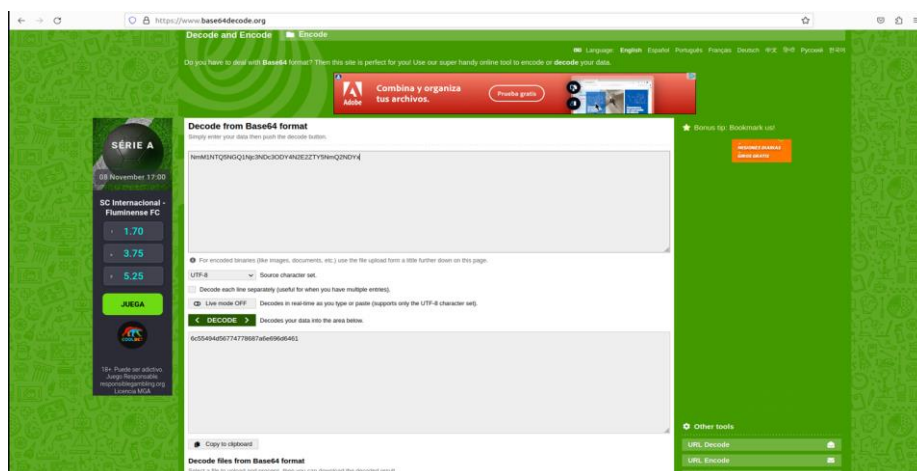
Haz Clic en “Forward” en el Proxy y verifica que se ha ingresado correctamente.



En el “Repeater” cambia las credenciales para ingresar con el usuario **admin**.

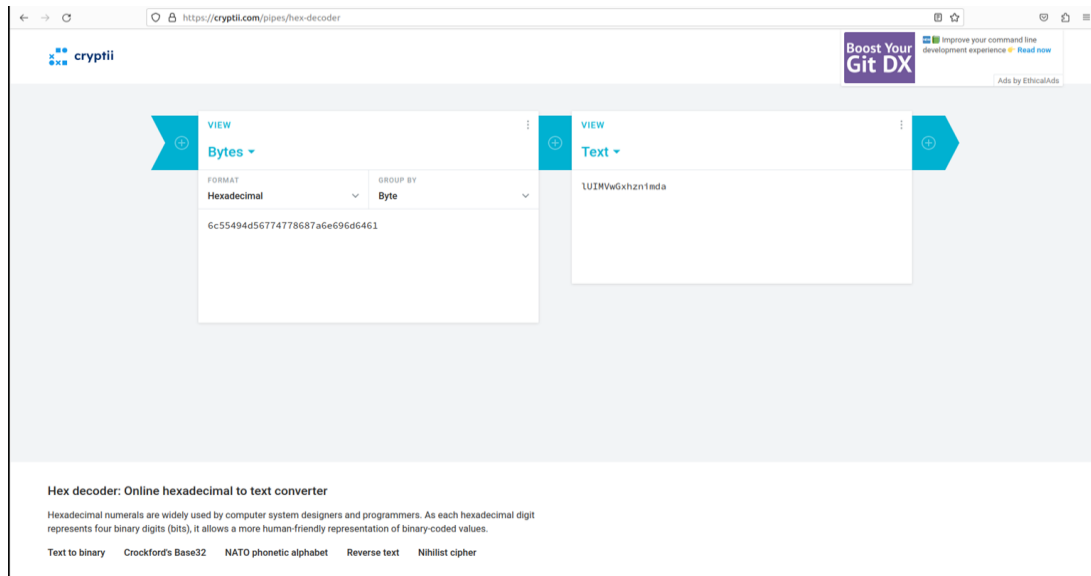


Decodifica la cookie.

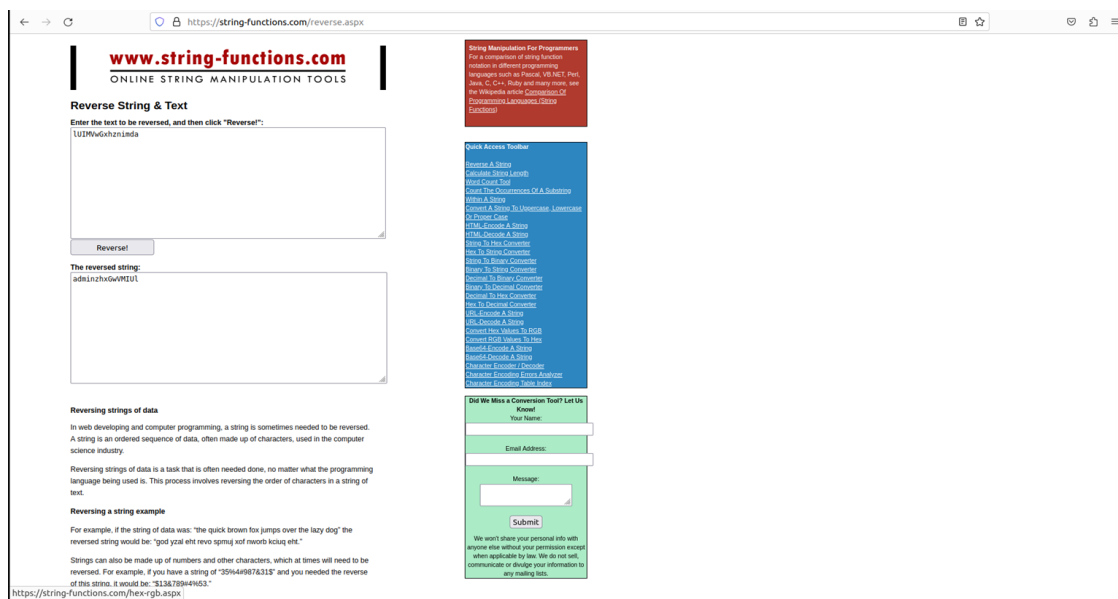


OWASP

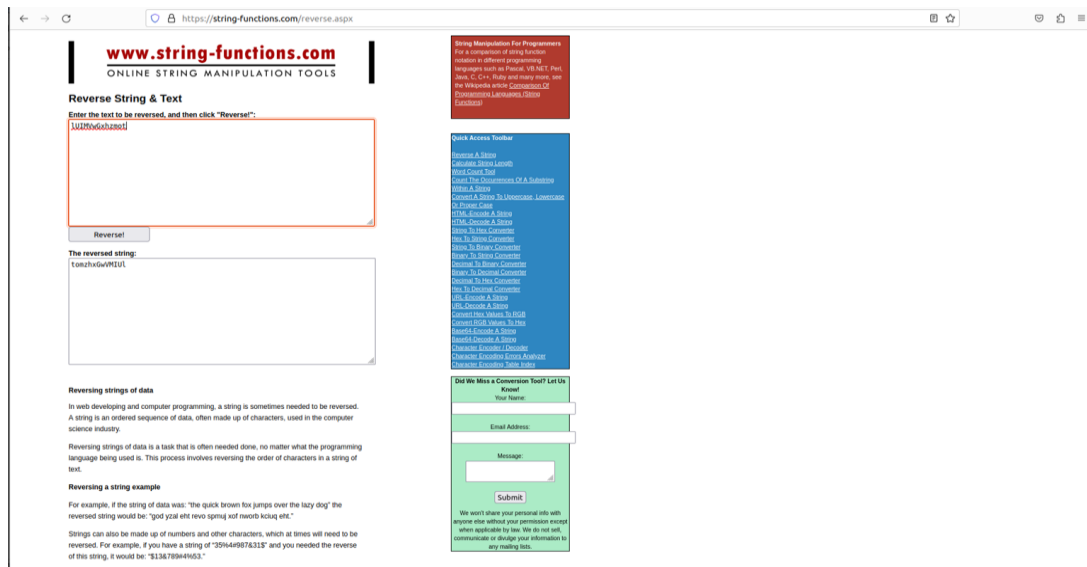
Convierte a cadena ASCII los códigos hexadecimales.



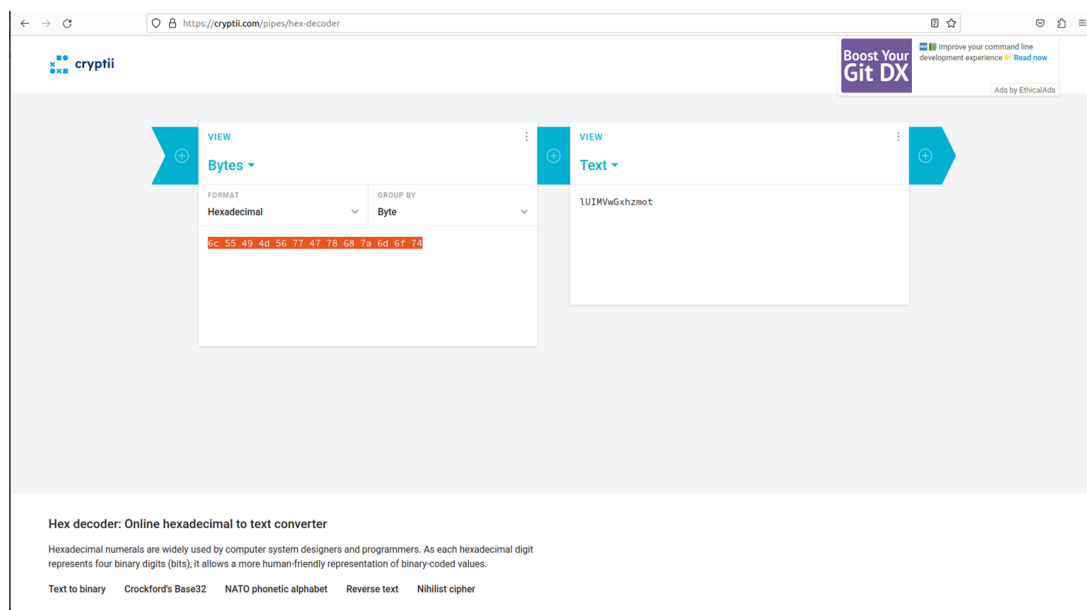
Convierte a reverse la cadena generada en el paso previo.



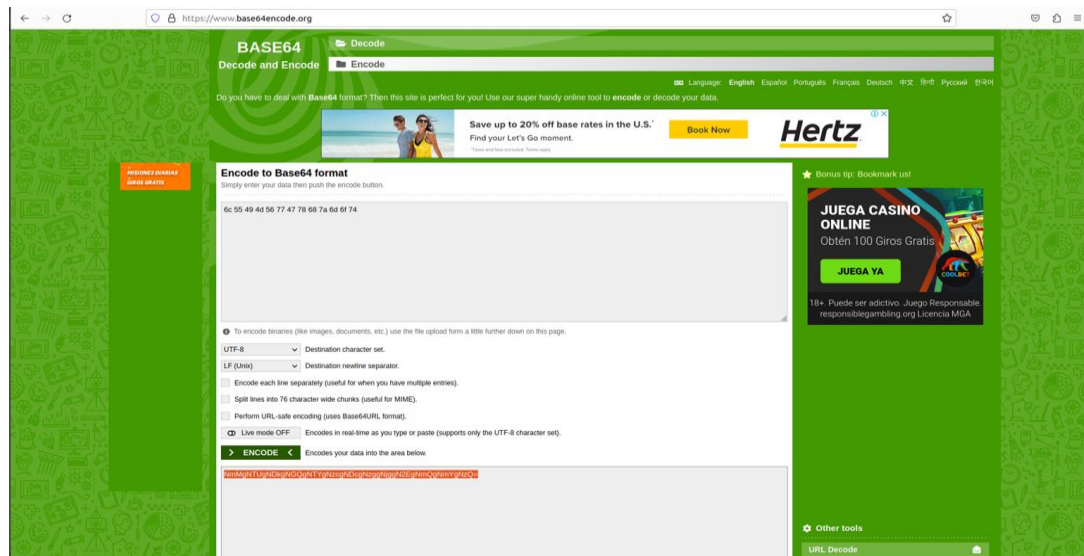
Finalmente podemos ganar acceso con otra cuenta si cambiamos el Cookie usando la cuenta tom. Obtén la cadena en reversa con la cuenta tom.



Decodifica para obtener los caracteres hexadecimales.



Obtén la Cookie para tom.



Finalmente prueba con Burp suite el login exitoso, con la cuenta tom.

