The source files of a sample application are attached as a .zip file.
Please unzip the zip file
in a directory. The code in the app.py file in the unzipped directory
contains several OWASP
Top 10:2021 vulnerabilities. Please review the code, identify contained
vulnerabilities, and
suggest possible mitigations to resolve the identified vulnerabilities.
Each area has to
include:

- What is the vulnerability?
- Why does the vulnerability arise (what is the reason for the vulnerability)?
- What is your mitigation suggestion to close the vulnerability?

Note 1: 10 areas are given below to write your answers. However, the
application may or
may not include 10 vulnerabilities.

Note 2: Applicants can run the sample application using docker with the
“docker-compose
up” command if necessary.

1)OWASP TOP 10:2021 NAME = A03:2021 – Injection

Code line 22 (statement = "SELECT username,password from users where
username = '"+username+"' and password='"+password+"'")

SQL injection

Payload: http://0.0.0.0:8000/login/'or 1=1--/''

Problem:

Since the 1=1 statement in the payload will always be true, the login
will be logged. The -- statement disables the rest of the query.

Solution:

Don’t trust any user input
Use whitelists, not blacklists
Employ verified mechanisms

2)OWASP TOP 10:2021 NAME = A02:2021-Cryptographic Failures

Insecure cryptography algorithm

Code line 68 (m = hashlib.md5(key.encode('utf-8')))
Code line 80   (m = hashlib.md5(key.encode('utf-8')))

Problem:

Using insecure cryptographic algorithms to hash passwords can be dangerous and exploitable. Algorithms such as MD2, MD3, MD4, MD5, SHA, SHA1_CRYPT, MD5_CRYPT are broken.
They can be used for hashing files only; they should not be used for password or any important credentials.

Solution:

Identify whether a hash is used for a password (check for permutations of password) or string and use SHA256 or a higher algorithm.
3)OWASP TOP 10:2021 NAME = A05:2021 – Security Misconfiguration

Hardcoded credentials in Python code

Code line 14
Code line 65
Code line 77
Code line 91

Problem:

The detected Python code has hardcoded credentials or secrets. This code could be used to break into the system if an an attacker or a malicious internal employee gains access it

Solution:

-Rather than hardcoding the credentials in the code itself, pass credentials using an environment variable
-Alternately, you can use a separate Python script imported at runtime but stored separately from your codebase
4)OWASP TOP 10:2021 NAME = A03:2021 – Injection

Insert SQL Injection

Problem:

There is an insert sql injection resulting from the following sql query

```
statement = "INSERT INTO tokens(token_value) VALUES ('"+token+"')"
```

Solution:

Use PDO and prepared queries.
Use prepared statements and parameterized queries.
These are SQL statements that are sent to and parsed by the database server separately from any parameters.
This way it is impossible for an attacker to inject malicious SQL

5)OWASP TOP 10:2021 NAME = A03:2021 – Injection

Reflected XSS

Scripts running in browser.There is an XSS vulnerability caused by the following code block.

```
----------------------------------------------------------------------------------------------------------------
session_id_cookie = request.cookies.get('session_id').split('.')
decrpyed_value = des_decrypt(session_id_cookie[1])
----------------------------------------------------------------------------------------------------------------
```

Problem:
Cookies stored directly

Solution:
Use HTML ONLY

Details Of The Solution:
If the HttpOnly flag is included in the HTTP response header, the cookie cannot be accessed through the client-side script.
As a result, even if a cross-site scripting (XSS) flaw exists, and a user accidentally accesses a link that exploits the flaw, the browser will not reveal the cookie to the third-party.

6)OWASP TOP 10:2021 NAME = A03:2021 – Injection

Problem:

The database can be accessed using 3 SQL injections.

```
-------------------------------------------------------------------------------------------------------------------------------
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: http://0.0.0.0:8000/login/'or 1=1--/'' AND 1692=1692 AND
'BOEQ'='BOEQ

    Type: error-based
    Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or
GROUP BY clause (GTID_SUBSET)
    Payload: http://0.0.0.0:8000/login/'or 1=1--/'' AND
GTID_SUBSET(CONCAT(0x7171787871,(SELECT
(ELT(6025=6025,1))),0x716a7a7671),6025) AND 'TOEX'='TOEX

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: http://0.0.0.0:8000/login/'or 1=1--/'' AND (SELECT 7923 FROM
(SELECT(SLEEP(5)))fjjz) AND 'INLa'='INLa
-----------------------------------------------------------------------------------------------------------------------------------
```

Solutions:

The preferred option is to use a safe API, which avoids using the interpreter entirely, provides a parameterized interface, or migrates to Object Relational Mapping Tools (ORMs).
Use positive server-side input validation. This is not a complete defense as many applications require special characters, such as text areas or APIs for mobile applications.For any residual dynamic queries, escape

special characters using the specific escape syntax for that interpreter.Use LIMIT and other SQL controls within queries to prevent mass disclosure of records in case of SQL injection.

Data obtained using SQL injection

username:admin
password:adminP@ssw0rd
username:johnny
password:t0rt0iss3

7)

A07:2021 – Identification and Authentication Failures

```
-----------------------------------------------------------------------------------------------
def login(username,password):
    dbinit.db_init()
    mydb = mysql.connector.connect(
        host="mysqldb",
        user="root",
        password="p@ssw0rd1",
        database="assignment"
    )
------------------------------------------------------------------------------------------------
```

Problem:

Connected to mysql directly as root user. The password for root user is very weak and commonly used password

Solution:

A strong and complex password should be used.