SECURITY AUDIT REPORT

Project Name: cybersec_task2

Audit Type: Defensive Security Assessment

Submitted by:

Name: ARJAV JAIN

Reg No. – 25BYB0108

Department: B.Tech in Cybersecurity

Submitted to:

VinnovateIT

1. Project Understanding

- **Purpose of the Project –**
  This project is a backend API designed for a simple e-commerce application. It handles user authentication, product management, and order processing. The application is built using Python and Flask and stores data in a database.
- **Functionality –**
  - User login and authentication
  - Product listing and management
  - Order creation and order handling
- **Exposed Attack Surface –**
  - Authentication APIs (login handling)
  - Product-related API endpoints
  - Order-related API endpoints
  - User input sent through API requests
  - Database access through backend logic

2. Code-Level Security Review

File: app.py –

Issue 1: Debug mode enabled

Why this is risky:

- When debug mode is enabled, the application may show internal errors and details. These details should not be visible when the application is used in real situations because they can expose sensitive information.

File: auth.py –

Issue 1: Weak password handling

Why this is risky:

- Passwords are not properly protected, so if the data is leaked, user accounts can be easily misused.

Issue 2: Hardcoded secret key

Why this is risky:

- Since the secret key is written in the code, anyone with access to the code can misuse it and create fake authentication tokens.

File: database.py –

Issue 1: Hardcoded database path

Why this is risky:

- The database path is hardcoded in the code, which makes it easier for attackers to locate sensitive data and reduces security.

Issue 2: Improper error handling

Why this is risky:

- Database errors are not handled properly, which can expose internal details and affect the stability of the application.

File: products.py –

Issue 1: No input validation

Why this is risky:

- The application does not validate user input, so invalid or harmful data can affect how the application works.

Issue 2: Missing authentication checks

Why this is risky:

- Authentication checks are missing, so unauthorized users can access product-related data.

File: orders.py –

Issue 1: Missing authorization checks

Why this is risky:

- Authorization checks are missing, so users can access orders that do not belong to them.

Issue 2: Trusting user input directly

Why this is risky:

- The application trusts user input directly, so attackers can manipulate order details.

3. Dependency & Supply Chain Risks

Dependencies Used:

- The project uses external Python libraries listed in requirements.txt.

Identified Risks:

- The libraries do not have fixed version numbers.
- Different versions of the same library can be installed at different times.

Why this is risky:

- New versions of libraries may contain bugs or security issues.
- If any external library becomes unsafe, the application can also be affected.
- Since the project depends on outside libraries, this creates a security risk

4. Configuration & Deployment Risks

Insecure Default Configuration

- Debug mode is enabled in the application.

Missing or Misconfigured Security Headers

- Basic security headers are not added in the application.

Weak Handling of Secrets

- Secret keys are written directly in the code instead of being stored safely.

Logging of Sensitive Information

- Error messages may show internal details of the application.

5.Threat Modeling

Potential Threat Actors

- External attackers who try to access the system from outside.
- Logged-in users who may misuse their access.

Possible Attack Paths

- Misuse of authentication tokens.
- Accessing orders without proper permission.
- Sending modified or incorrect data through the API.

Risk Ranking

- Authentication and authorization issues - High
- Input validation issues - Medium
- Configuration and dependency issues - Medium
- Documentation-related issues - Low

6. Recommendations

High Priority

- Turn off debug mode when the application is used in real environments.
- Improve how passwords are handled and make login more secure.
- Remove secret keys from the code and store them in a safer way.

Medium Priority

- Check and validate all user inputs before using them.
- Make sure users can only access data they are allowed to see.
- Add basic security headers to improve browser safety.

Low Priority

- Improve documentation by adding security-related warnings.
- Use fixed versions for external libraries to avoid unexpected issues.

7. Conclusion -

The application has several security issues, including some high and medium level risks. Because of these issues, the application is not safe to be used in a real production environment at the moment.

If the identified issues are fixed, the security of the application will improve and it will become more reliable for users.