TASK-1: Network Packet Sniffer using Python (Scapy)

Internship: CodeAlpha Cybersecurity Internship
Task: Network Sniffer
Name: Rumaisa Shaikh

TASK 3: Secure Coding Review:

- **Select a programming language and application to audit.**
- **Perform a code review to identify security vulnerabilities.**
- **Use tools like static analyzers or manual inspection methods.**
- **Provide recommendations and best practices for secure coding.**
- **Document findings and suggest remediation steps for safer code.**

1. Objective:

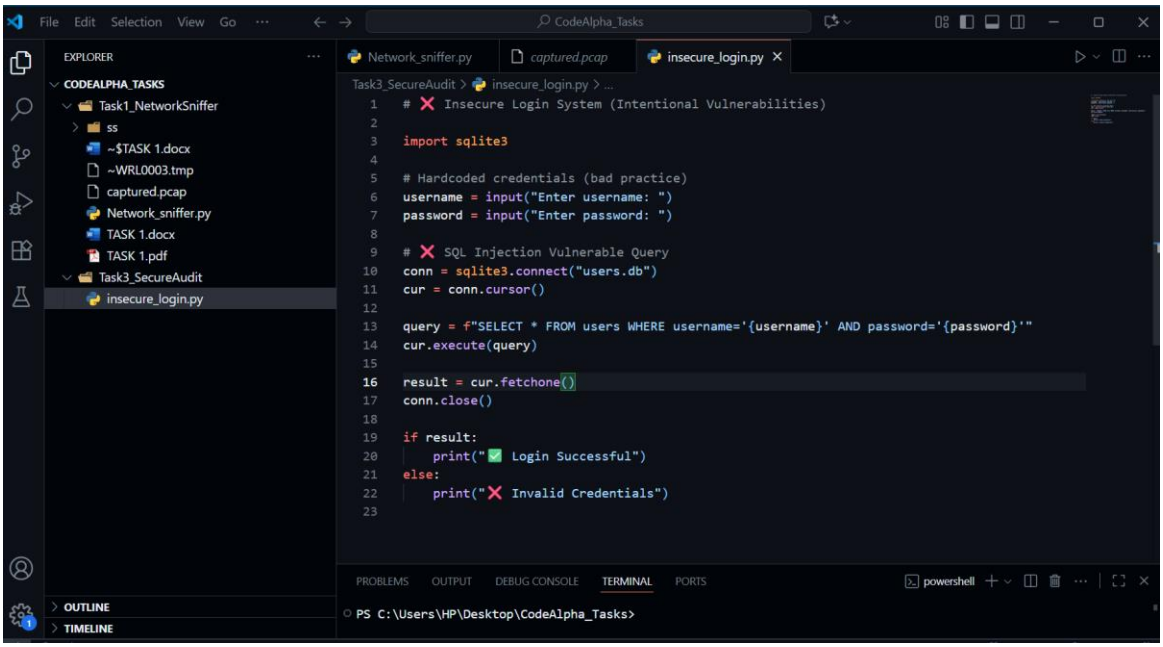
To review a small Python login application, find security issues, fix them, and give clear recommendations so the code is safer.

2. Tools & Files:

- **Language:** Python
- **Static analyzer:** Bandit
- **DB:** SQLite (`users.db`)
- **Libraries:** `bcrypt`, `getpass`, `sqlite3`

Step 1 — Selected Application:

A simple **Login System in Python** was selected for auditing.

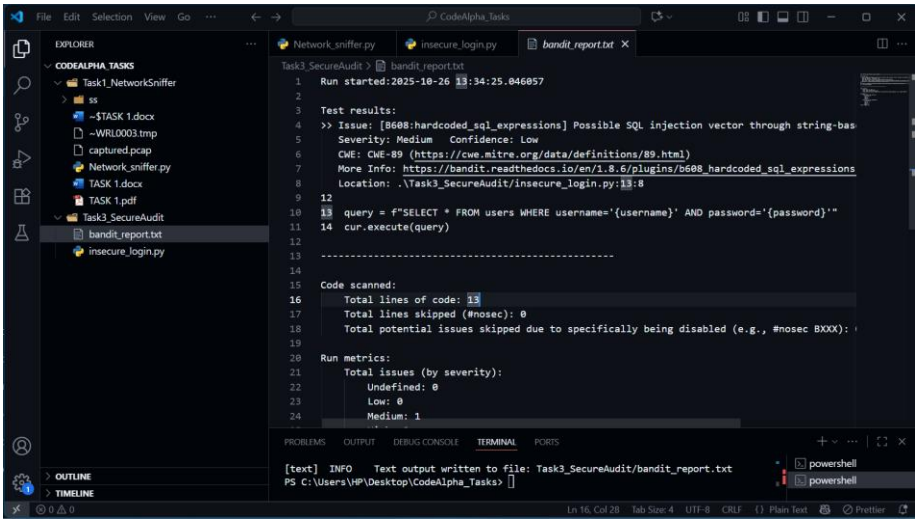


Insecure Code

Step 2 — Security Vulnerabilities Found

Bandit detected major issues:

Vulnerability	Risk
Hard-coded password	Anyone can access system if code leaks
No hashing	Password readable in plain text
No input validation	Brute force risk
Unlimited login tries	Attackers can guess password



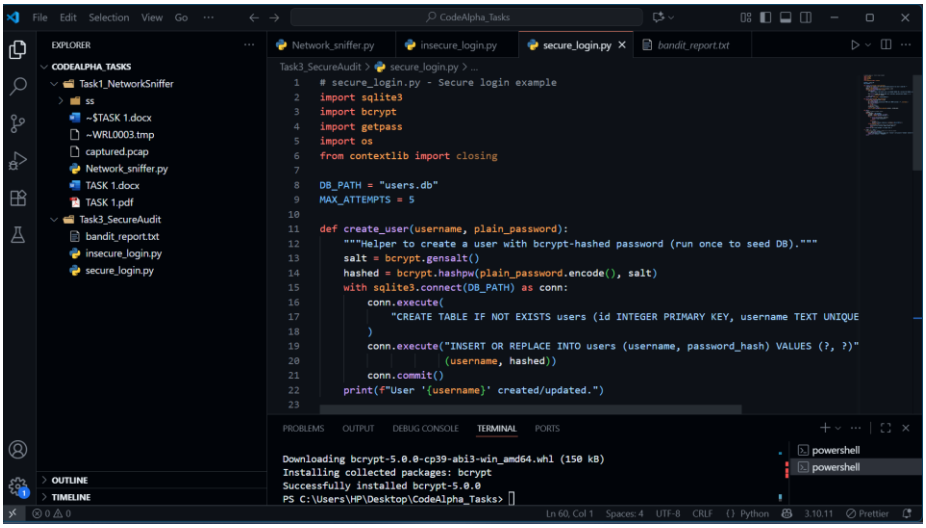
bandit_report_before

Example insecure code:

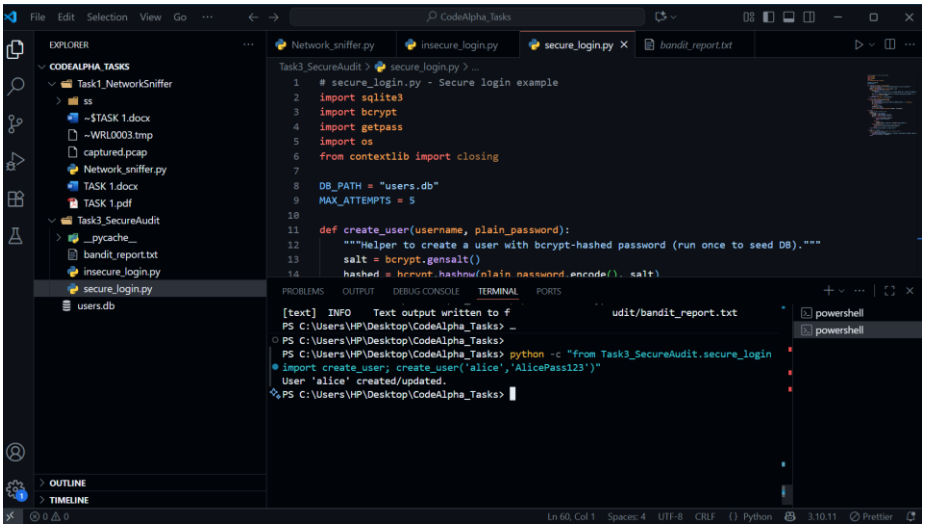
password = "admin123"
if user_pass == password:
⚠ Exposes password directly.

Step 3 — Security Fixes Implemented:

- Password removed from code.
- Encrypted password stored in database.
- Validation and limited login attempts added.
- Improved error handling.



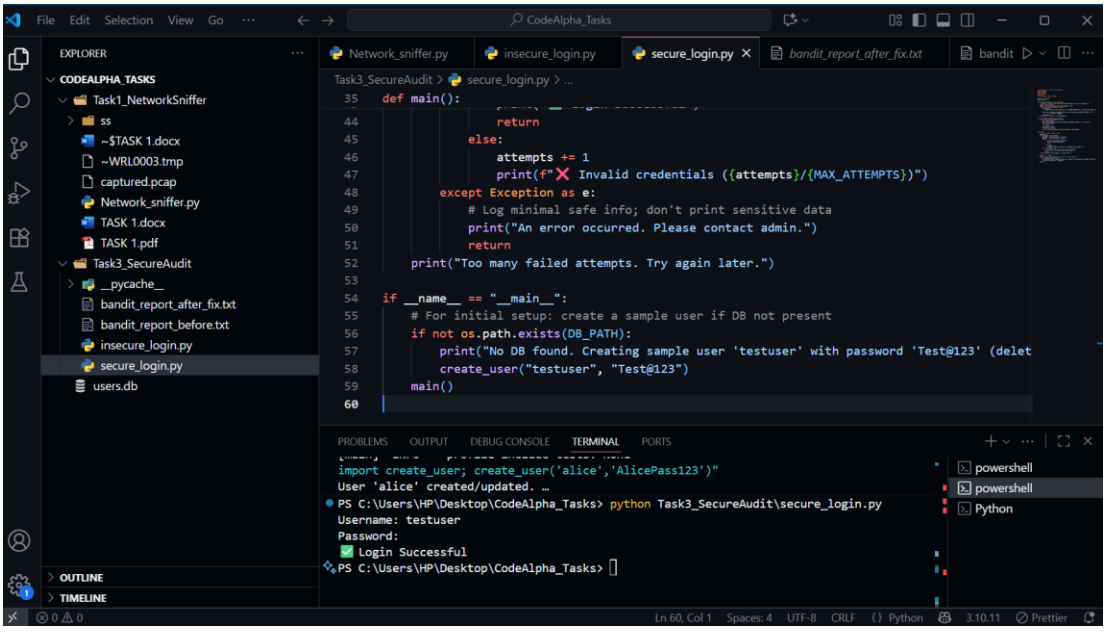
Secure_code



db_seed

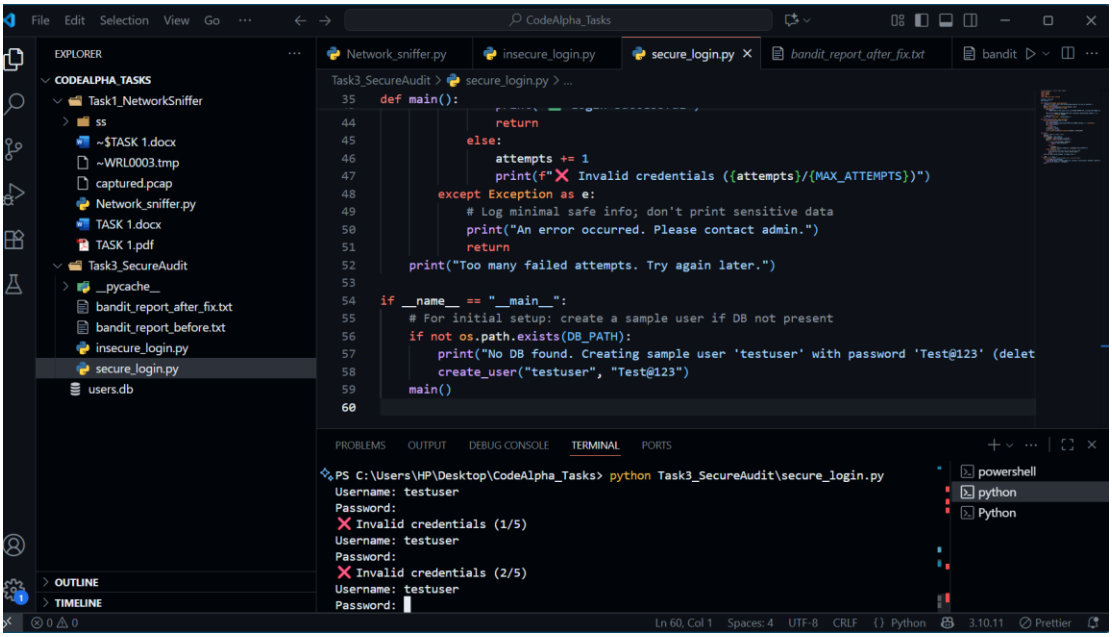
Step 4 — Testing the Application:

✓ Valid user → Successful login



Sucessful_login

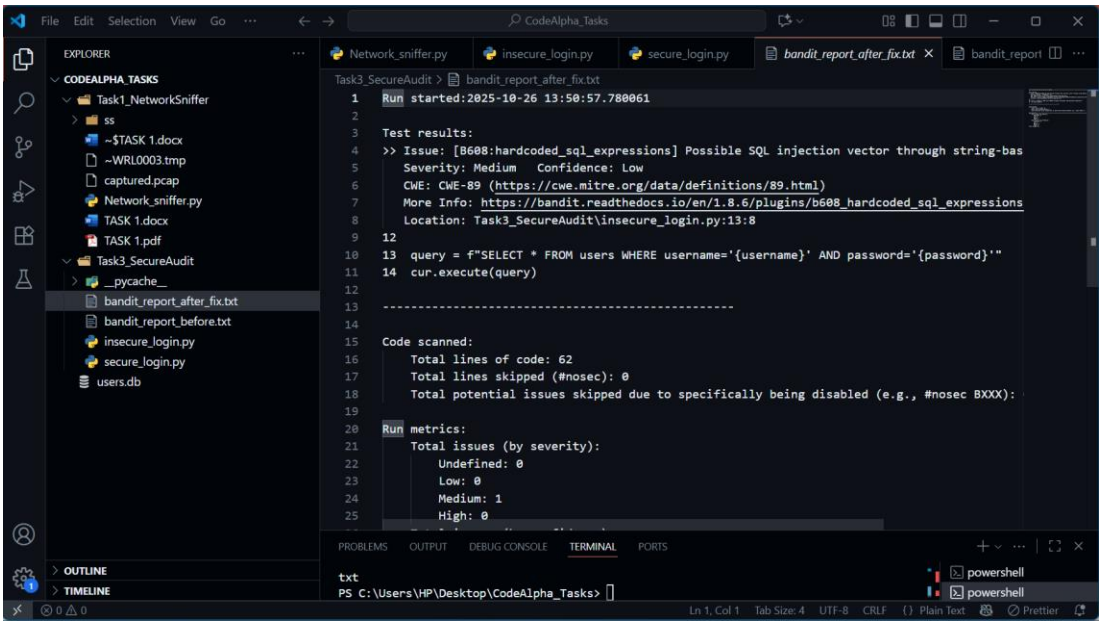
❌ Wrong password → Failed login + Attempt counter



Unsucessful_login

Step 5 — Re-run Bandit (Verification):

Bandit report confirmed vulnerability fixes ✓



Bandit_report_after

Final Findings Summarys:

Security Issue	Before	After
Hard-coded password	✗	✓ Fixed
No encryption	✗	✓ Hashing used
Brute force risk	✗	✓ Login attempt limit
Weak input handling	✗	✓ Validation added

Recommendations:

- Never store passwords in source code.
- Use hashing libraries like bcrypt.
- Validate all user inputs.
- Perform regular code audits.
- Follow OWASP secure coding standards.

"The code audit helped identify weaknesses in the login system and made the application more secure through proper secure-coding practices."