Task 3 Secure Coding Review

simple **Python web application using Django** to illustrate the process of reviewing for security vulnerabilities.

```python
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login
from django.http import HttpResponse
from .models import User

def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('dashboard')
        return HttpResponse('Invalid credentials', status=401)
    return render(request, 'login.html')
```

Review :

1-**Input Validation**

-The code does not validate the format of the username and password inputs.

-Implement input validation to ensure the inputs conform to expected patterns, helping prevent injection attacks.

```python
import re

def is_valid_username(username):
    return re.match(r'^[a-zA-Z0-9_]+$', username) is not None
```

2-**Error Handling**

-The application returns a generic error message, but it might still give hints about valid usernames or passwords.

-Log the failed login attempts for monitoring while returning a generic message to the user. Consider logging the attempt with the IP address for better tracking.

```python
import logging

logging.basicConfig(level=logging.INFO)
def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        logging.warning(f"Failed login attempt for username: {username} from IP: {request.META['REMOTE_ADDR']}")
```

3-**Password Handling**

-Vulnerability: The code uses Django's built-in authenticate method, which is good; however, ensure the passwords are stored securely with proper hashing.

-Recommendation: Verify that Django is configured to use strong password hashing algorithms (like PBKDF2, Argon2) in your settings.

4-**Logging and Monitoring**

-Vulnerability: The application does not implement sufficient logging for security events.

-Recommendation: Set up logging to track important actions (like login attempts) and consider using a logging framework to manage different log levels.

5-**Dependencies and Security Updates**

-The code does not mention how to handle dependencies, which can be a risk if outdated libraries are used.

-Regularly review and update dependencies. Use tools like safety and pip-audit to scan for vulnerabilities in installed packages.

```
PS D:\Cyber Security code alpha> & C:/Users/PC/AppData/Local/Programs/Python/Python312/python.exe "d:/Cyber Security code alpha/task3.py"
PS D:\Cyber Security code alpha> pip-audit
Found 2 known vulnerabilities in 2 packages
Name    Version  ID                Fix Versions
------- -------- ----------------- ------------
certifi 2024.6.2 GHSA-248v-346w-9cwc 2024.7.4
urllib3 2.2.1    GHSA-34jh-p97f-mpxf 1.26.19,2.2.2
PS D:\Cyber Security code alpha>
```

Tools for Static Code Analysis :

Bandit: Use Bandit to find common security issues in Python code

```
PS D:\Cyber Security code alpha> bandit D:\Cyber Security code alpha/task3
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
[main]  INFO    running on Python 3.12.0
Run started:2024-11-02 15:39:29.112645

Test results:
        No issues identified.

Code scanned:
        Total lines of code: 0
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
        Total issues (by confidence):
                Undefined: 0
                Low: 0
                Medium: 0
                High: 0
```

Code with Recommendations :

```python
import re
import logging
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login
from django.http import HttpResponse
logging.basicConfig(level=logging.INFO)
def is_valid_username(username):
    return re.match(r'^[a-zA-Z0-9_]+$', username) is not None
def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        if not is_valid_username(username):
            logging.warning(f"Invalid username attempt: {username} from IP: {request.META['REMOTE_ADDR']}")
            return HttpResponse('Invalid credentials', status=401)
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('dashboard')
        logging.warning(f"Failed login attempt for username: {username} from IP: {request.META['REMOTE_ADDR']}")
        return HttpResponse('Invalid credentials', status=401)
    return render(request, 'login.html')
```