OWASP Vulnerabilities

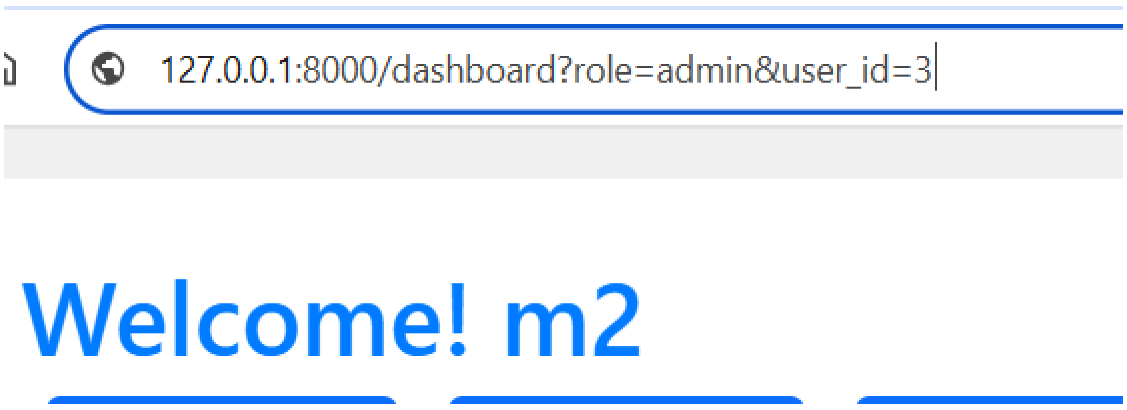
A01: Broken Access Control

Missing Authentication for Critical Functions

Description

Unauthorized users can access resources or functionalities meant for higher-privileged users, such as admin functionalities.

Image of Benign Request



127.0.0.1:8000/dashboard?role=admin&user_id=3

Welcome! m2

Exploitation Steps

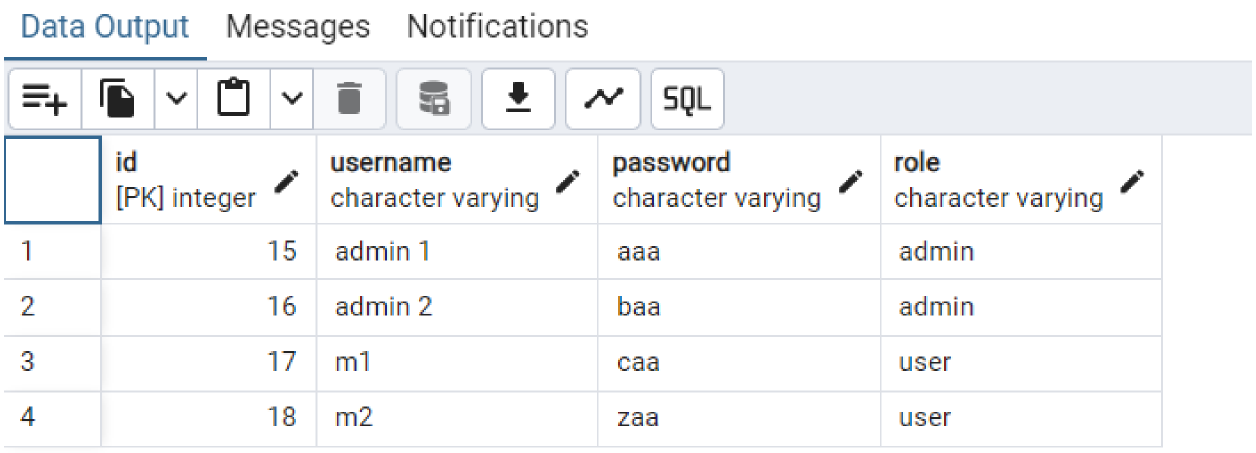
1. **Log in as a low-privileged user.**
2. **Attempt to access the admin dashboard URL directly (/dashboard?role=admin&user_id=3).**
3. **Observe if access is granted without proper authentication or authorization.**
4. **Impact: Unauthorized access to admin's functionalities.**

A02: Cryptographic Failures

Missing Encryption of Sensitive Data

Description Sensitive data, such as passwords, are improperly stored or transmitted without encryption.

Image of Benign Request



	id [PK] integer	username character varying	password character varying	role character varying
1	15	admin 1	aaa	admin
2	16	admin 2	baa	admin
3	17	m1	caa	user
4	18	m2	zaa	user

Exploitation Steps

1. Capture login request using a proxy tool.
2. Check the database for improper password storage.
3. Attempt to intercept and re-play the request.

Impact: Exposure of sensitive data to attackers.

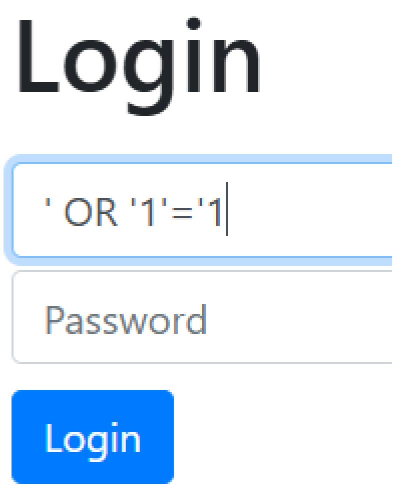
A03: Injection

1. **SQL Injection:**

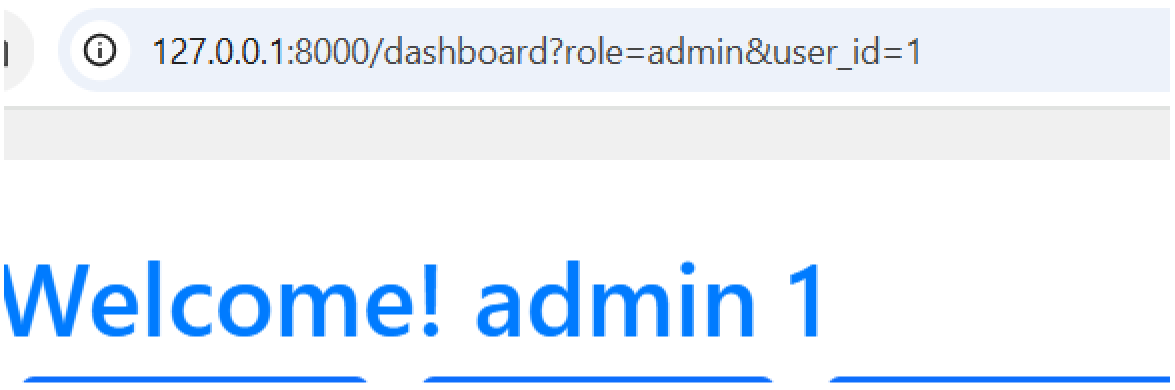
Description

User input is improperly sanitized, allowing malicious SQL queries to be executed.

Image of Benign Request



Then directs to the admin's dashboard:



Exploitation Steps

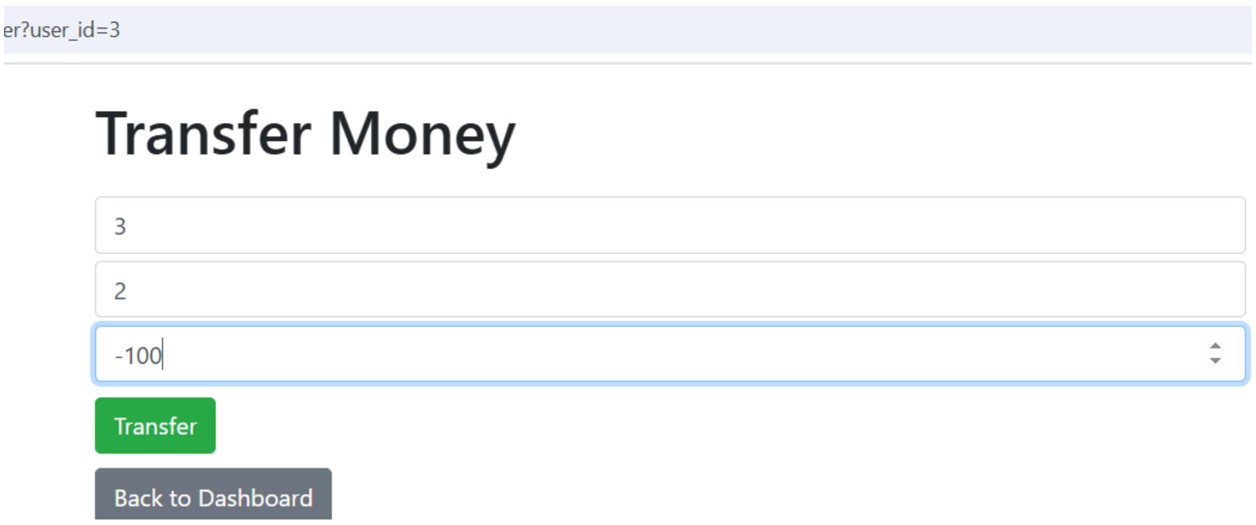
1. Navigate to the login or transaction page.
2. Enter a payload like ' OR '1'='1 in input fields.
3. Observe if the response grants unauthorized access or exposes data.
4. Impact: Unauthorized access to sensitive information.

A04: Insecure Design

1. **Business Logic Flaw (Negative Transfers)**

Description:
The application fails to validate that transfer amounts must be positive numbers, allowing attackers to steal funds by submitting negative values. This is a critical business logic flaw that violates the intended behavior of the banking system.

Exploitation Steps:

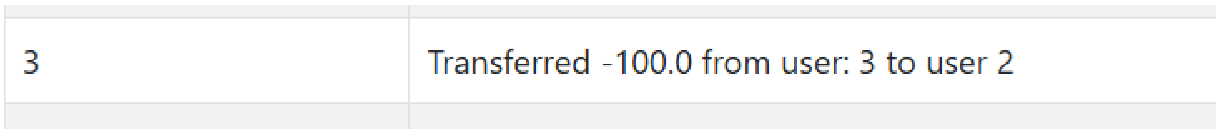

er?user_id=3

Transfer Money

3

2

-100



3	Transferred -100.0 from user: 3 to user 2

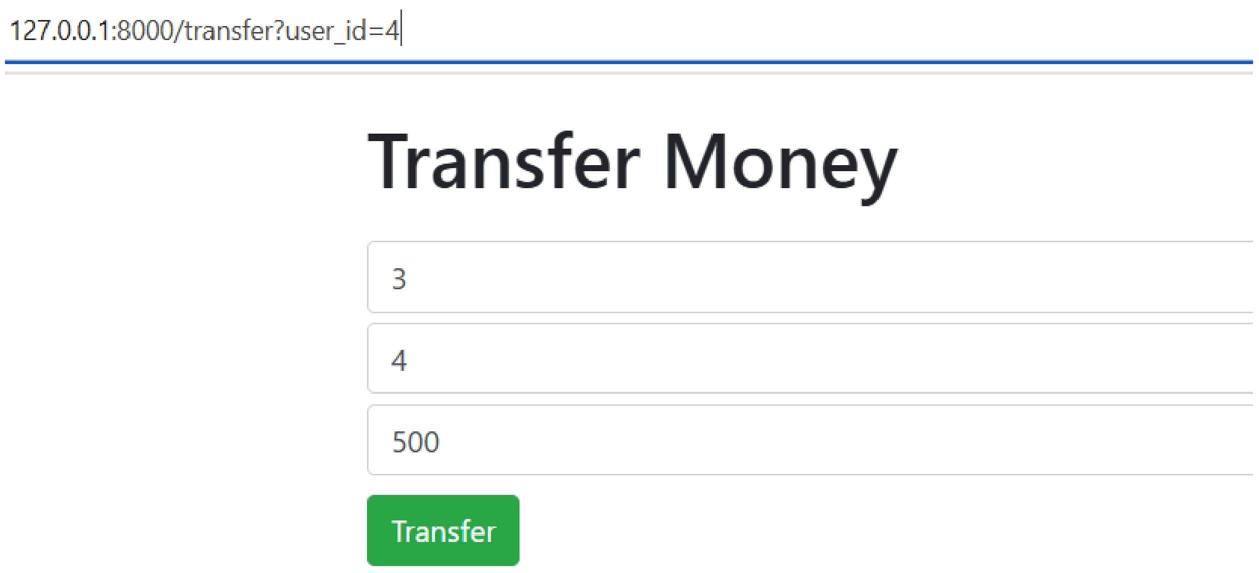
Impact:
The attacker’s account balance **increases by $100**, while the victim’s balance decreases

2. Reliance on Untrusted Inputs in a Security Decision

Description

Security settings are not properly configured, leaving the application exposed to attacks.

Image of Benign Request


127.0.0.1:8000/transfer?user_id=4

Transfer Money

3

4

500

Exploitation Steps

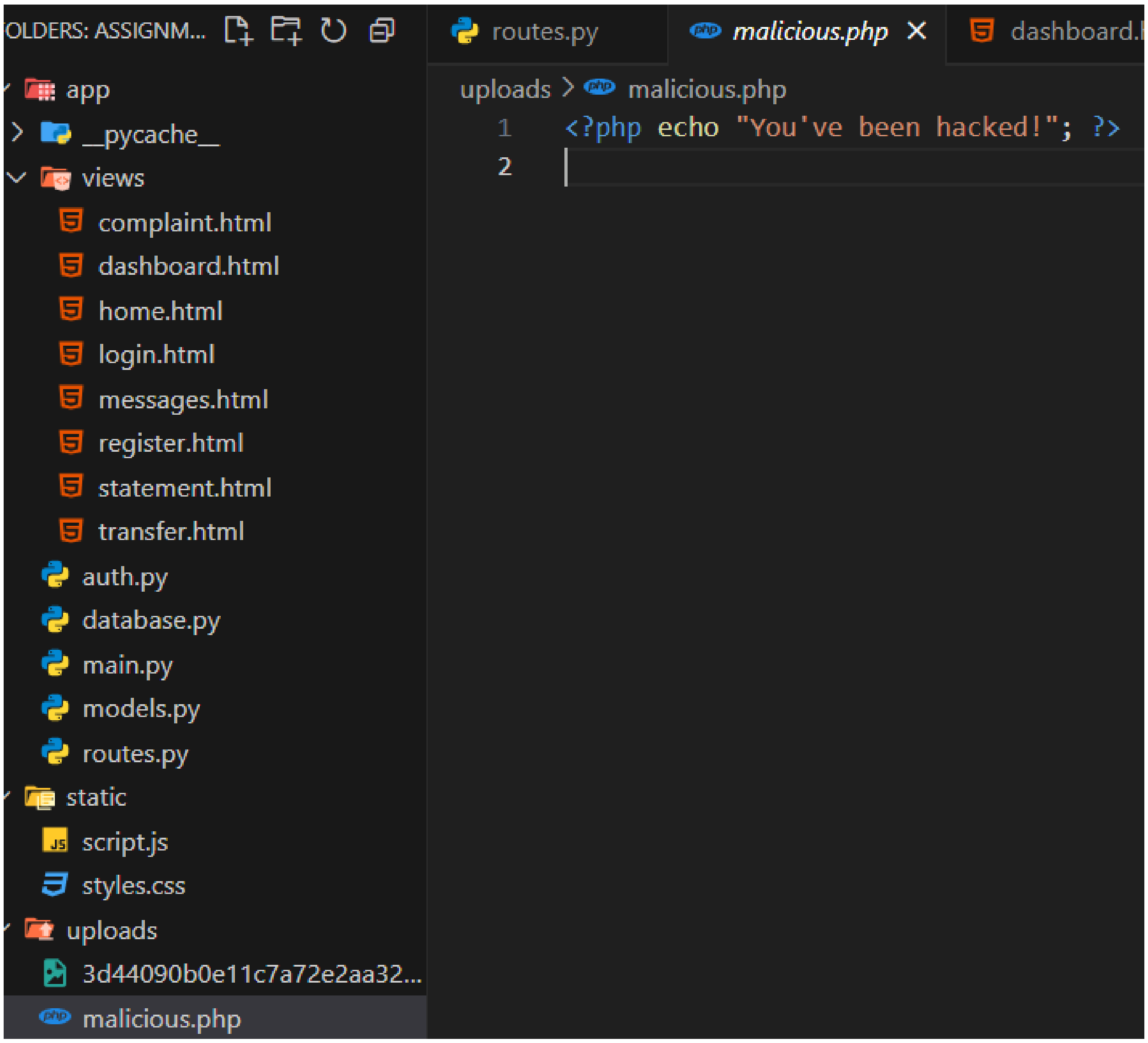
3. Log in as a regular user (e.g., m1).
4. Navigate the transfer money page and inspect the request.
5. Modify the sender_id to a different user's ID (e.g., another user).
6. Submit the modified request and observe the result.

A05: Security Misconfiguration

Unrestricted Upload of File with Dangerous Type

Description:
The application allows file uploads without proper validation, enabling attackers to upload malicious files like scripts or executables.

Image of benign request:



```php
<?php echo "You've been hacked!"; ?>

```

Exploitation Steps:

1. Navigate to the complaints or file upload page.
2. Upload a file with a dangerous extension (e.g., .php, .exe).
3. Try accessing the uploaded file directly via the application.
4. Observe if the file gets executed on the server.

Impact:
An attacker could gain remote code execution or manipulate server files

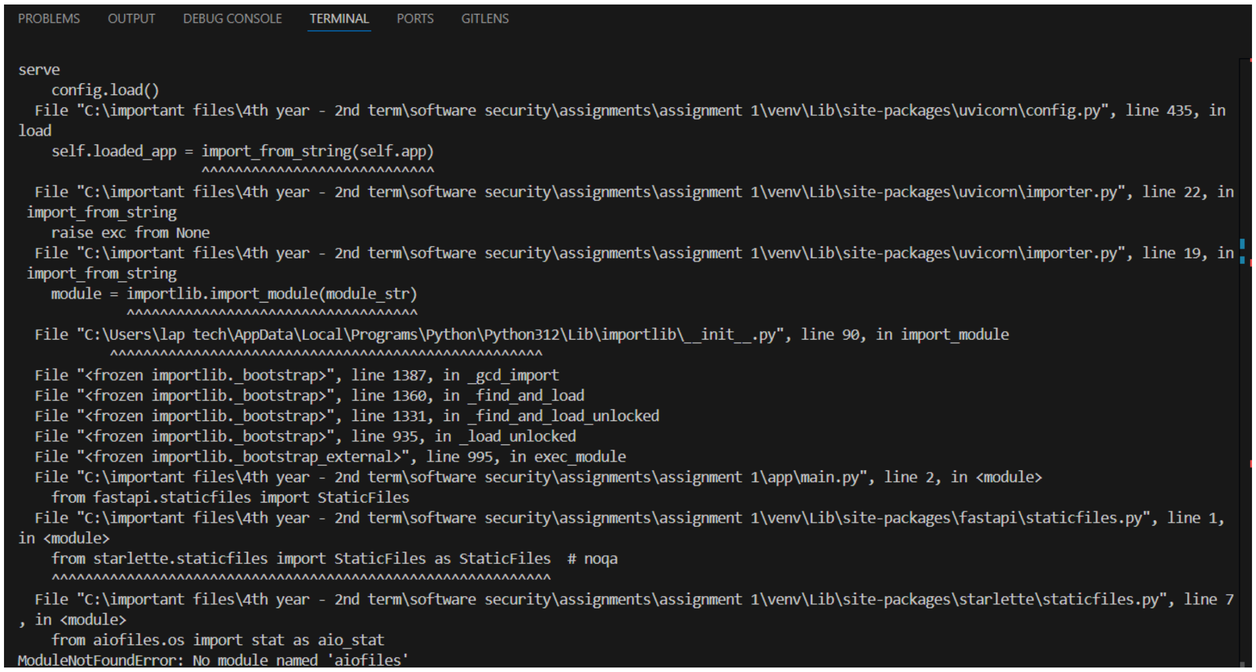
A06: Vulnerable and Outdated Components

Description:

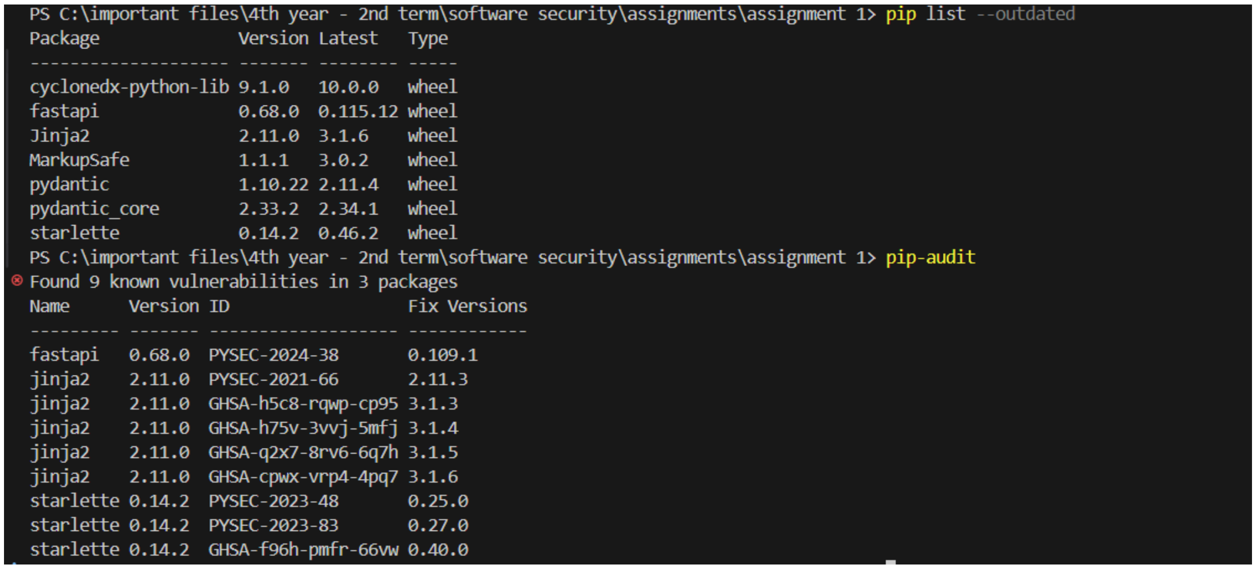
The application uses outdated versions of critical libraries (Jinja2 2.11.0, FastAPI 0.68.0) with known CVEs, exposing the system to:

- Server-Side Template Injection (SSTI) → Remote Code Execution (RCE)
- Open redirect attacks
- Security bypass vulnerabilities

Image of benign request:



```
serve
    config.load()
  File "C:\important files\4th year - 2nd term\software security\assignments\assignment 1\venv\Lib\site-packages\uvicorn\config.py", line 435, in
load
    self.loaded_app = import_from_string(self.app)
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\important files\4th year - 2nd term\software security\assignments\assignment 1\venv\Lib\site-packages\uvicorn\importer.py", line 22, in
 import_from_string
    raise exc from None
  File "C:\important files\4th year - 2nd term\software security\assignments\assignment 1\venv\Lib\site-packages\uvicorn\importer.py", line 19, in
 import_from_string
    module = importlib.import_module(module_str)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\lap tech\AppData\Local\Programs\Python\Python312\Lib\importlib\__init__.py", line 90, in import_module
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "<frozen importlib._bootstrap>", line 1387, in _gcd_import
  File "<frozen importlib._bootstrap>", line 1360, in _find_and_load
  File "<frozen importlib._bootstrap>", line 1331, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 935, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 995, in exec_module
  File "C:\important files\4th year - 2nd term\software security\assignments\assignment 1\app\main.py", line 2, in <module>
    from fastapi.staticfiles import StaticFiles
  File "C:\important files\4th year - 2nd term\software security\assignments\assignment 1\venv\Lib\site-packages\fastapi\staticfiles.py", line 1,
in <module>
    from starlette.staticfiles import StaticFiles as StaticFiles  # noqa
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\important files\4th year - 2nd term\software security\assignments\assignment 1\venv\Lib\site-packages\starlette\staticfiles.py", line 7
, in <module>
    from aiofiles.os import stat as aio_stat
ModuleNotFoundError: No module named 'aiofiles'
```



```
PS C:\important files\4th year - 2nd term\software security\assignments\assignment 1> pip list --outdated
Package             Version Latest   Type
------------------- ------- -------- -----
cyclonedx-python-lib 9.1.0   10.0.0   wheel
fastapi             0.68.0  0.115.12 wheel
Jinja2              2.11.0  3.1.6    wheel
MarkupSafe          1.1.1   3.0.2    wheel
pydantic            1.10.22 2.11.4   wheel
pydantic_core       2.33.2  2.34.1   wheel
starlette           0.14.2  0.46.2   wheel
PS C:\important files\4th year - 2nd term\software security\assignments\assignment 1> pip-audit
⊗ Found 9 known vulnerabilities in 3 packages
Name      Version ID                  Fix Versions
--------- ------- ------------------- ------------
fastapi   0.68.0  PYSEC-2024-38       0.109.1
jinja2    2.11.0  PYSEC-2021-66       2.11.3
jinja2    2.11.0  GHSA-h5c8-rqwp-cp95 3.1.3
jinja2    2.11.0  GHSA-h75v-3vvj-5mfj 3.1.4
jinja2    2.11.0  GHSA-q2x7-8rv6-6q7h 3.1.5
jinja2    2.11.0  GHSA-cpwx-vrp4-4pq7 3.1.6
starlette 0.14.2  PYSEC-2023-48       0.25.0
starlette 0.14.2  PYSEC-2023-83       0.27.0
starlette 0.14.2  GHSA-f96h-pmfr-66vw 0.40.0
```

Exploitation Steps:

1. Review requirements.txt
2. Identify outdated packages: pip list --outdated
3. Check for known CVEs in these versions

1. Identify Vulnerabilities: pip-audit

CVEs Found:

- CVE-2020-28493 (Jinja2 XSS → RCE)
- CVE-2021-33203 (FastAPI Open Redirect)

Impact:

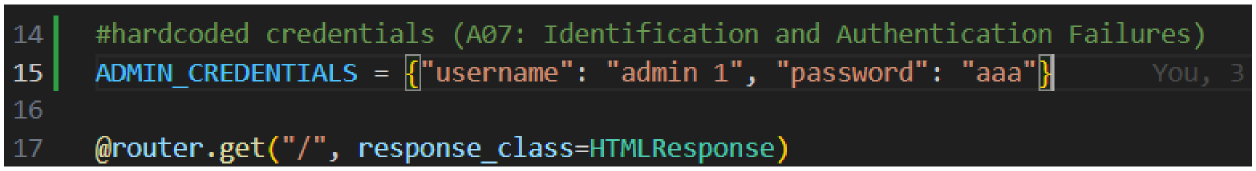
1. This dependency conflict crashed the application

2. Required manual dependency resolution

A07: Identification and Authentication Failures

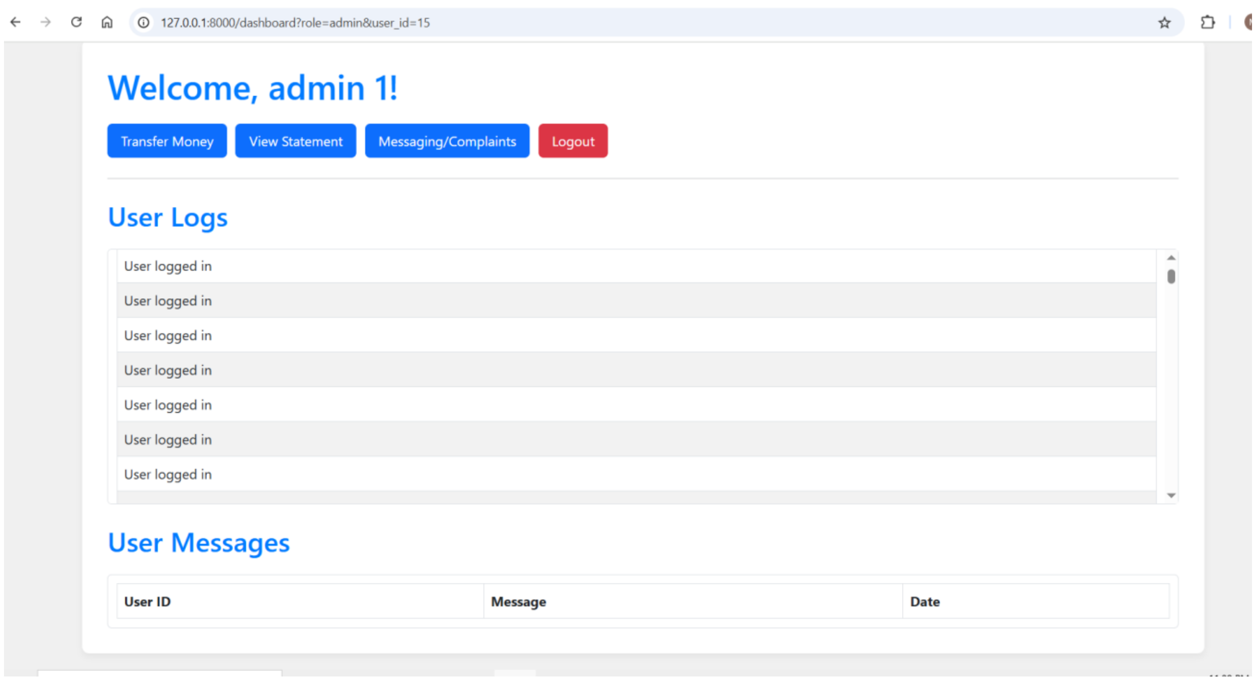
Use of Hardcoded Credentials

Description:
The application uses hardcoded credentials in source code, which can be exploited if discovered.



```
#hardcoded credentials (A07: Identification and Authentication Failures)
ADMIN_CREDENTIALS = {"username": "admin 1", "password": "aaa"}        You, 3

@router.get("/", response_class=HTMLResponse)
```



127.0.0.1:8000/dashboard?role=admin&user_id=15

Welcome, admin 1!

User Logs

User logged in
User logged in
User logged in
User logged in
User logged in
User logged in
User logged in

User Messages

User ID	Message	Date

Exploitation Steps:

1. Inspect application source code, configuration files, or decompiled binaries.
2. Look for hardcoded usernames and passwords.
3. Attempt to log in using the credentials discovered.

Impact:
An attacker could gain unauthorized access to the system, leading to data breaches or privilege escalation

---

A08: Software and Data Integrity Failures

Description:

The application lacks proper integrity checks for uploaded files, allowing potential tampering with critical data. The file upload feature accepts executable files without validation.

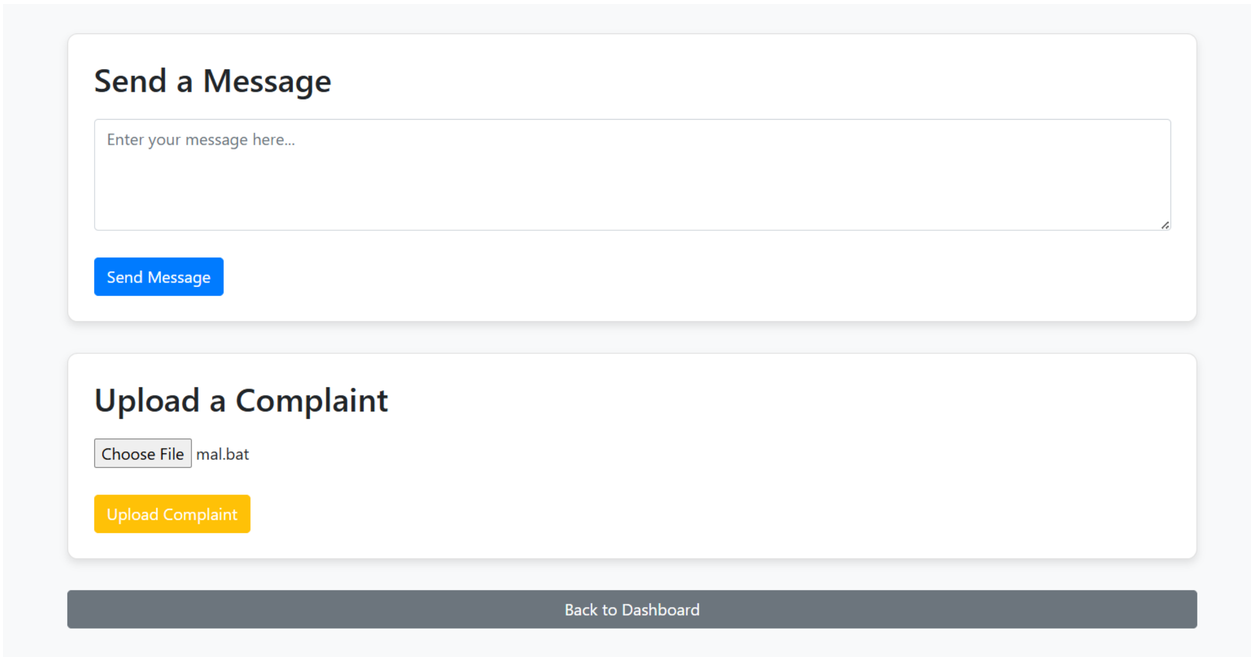
Exploitation Steps:

1. Create a malicious `.bat` file containing:

```
@echo off
whoami > C:\temp\compromise.txt
net user attacker P@ssw0rd123 /add
net localgroup administrators attacker /add
```

2. Upload the file through the complaints form
3. The server saves it without integrity checks
4. An attacker finds a way to execute the file (or it auto-executes)
5. The commands create a new admin user on the system

Image of benign request:

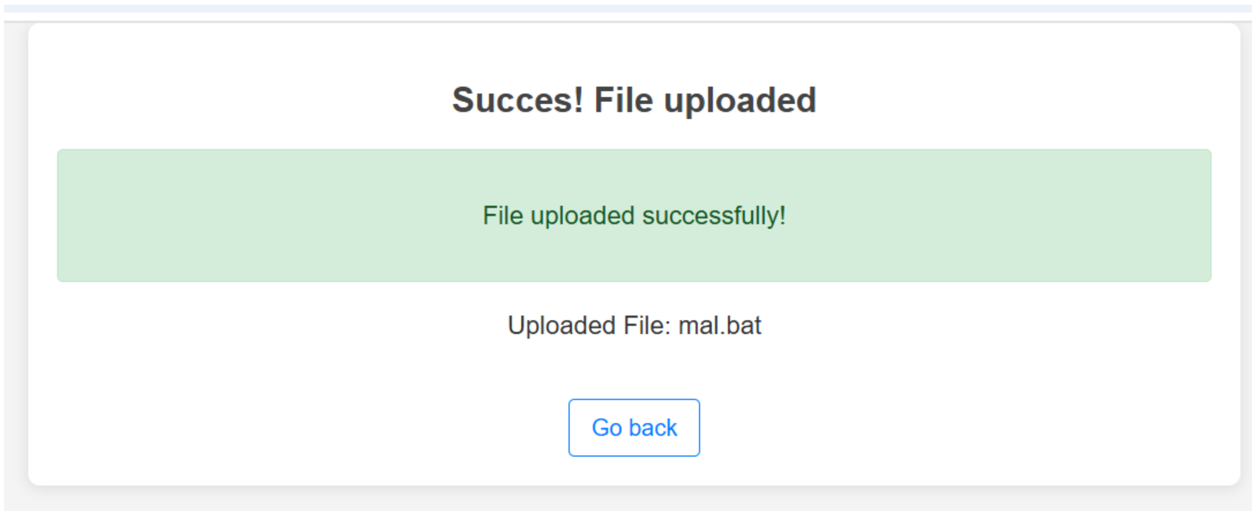


Send a Message

Enter your message here...

Upload a Complaint

Choose File mal.bat


Succes! File uploaded

File uploaded successfully!

Uploaded File: mal.bat

Impact:
Complete server compromise through arbitrary command execution and privilege escalation.

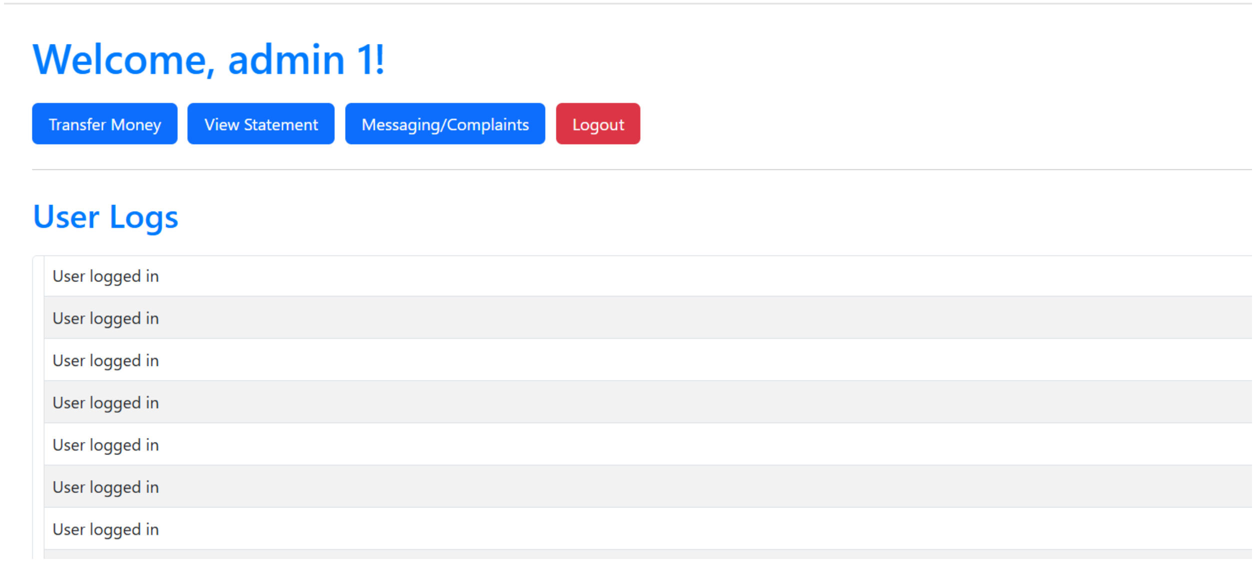
A09: Security Logging and Monitoring Failures

Category: Security Logging and Monitoring Failures
Description:
The application logs insufficient security events and fails to monitor for suspicious activities such as which users logged in, allowing attacks to go undetected.

- No username recorded
- No IP address logged
- No success/failure status

- Inability to detect brute force attacks
- No timestamp in the log entry

Image of benign request:



Welcome, admin 1!

User Logs

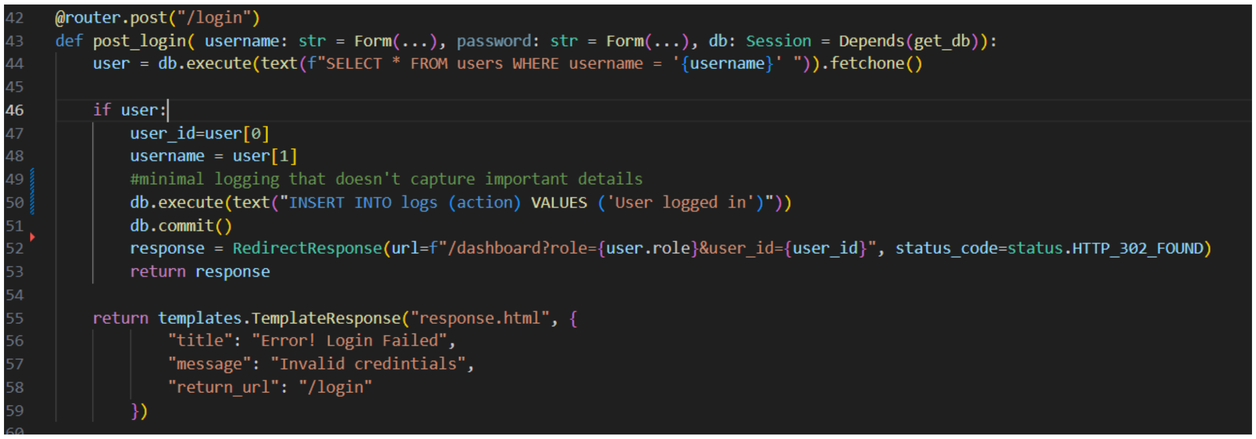
User logged in
User logged in
User logged in
User logged in
User logged in
User logged in
User logged in

Exploitation Steps:

1. **Perform multiple failed login attempts**
2. **Attempt SQL injection**
3. **Check if these events are properly logged**
4. **Verify if alerts would be generated**

Impact:

Inability to detect and respond to attacks in progress.



```
@router.post("/login")
def post_login( username: str = Form(...), password: str = Form(...), db: Session = Depends(get_db)):
    user = db.execute(text(f"SELECT * FROM users WHERE username = '{username}' ")).fetchone()

    if user:
        user_id=user[0]
        username = user[1]
        #minimal logging that doesn't capture important details
        db.execute(text("INSERT INTO logs (action) VALUES ('User logged in')"))
        db.commit()
        response = RedirectResponse(url=f"/dashboard?role={user.role}&user_id={user_id}", status_code=status.HTTP_302_FOUND)
        return response

    return templates.TemplateResponse("response.html", {
            "title": "Error! Login Failed",
            "message": "Invalid credintials",
            "return_url": "/login"
        })
```

A10: Server-Side Request Forgery (SSRF)

Description:
The application fetches user-supplied URLs for profile pictures without validation, allowing attackers to make requests to internal systems, cloud metadata services, or restricted endpoints.

Exploitation Steps

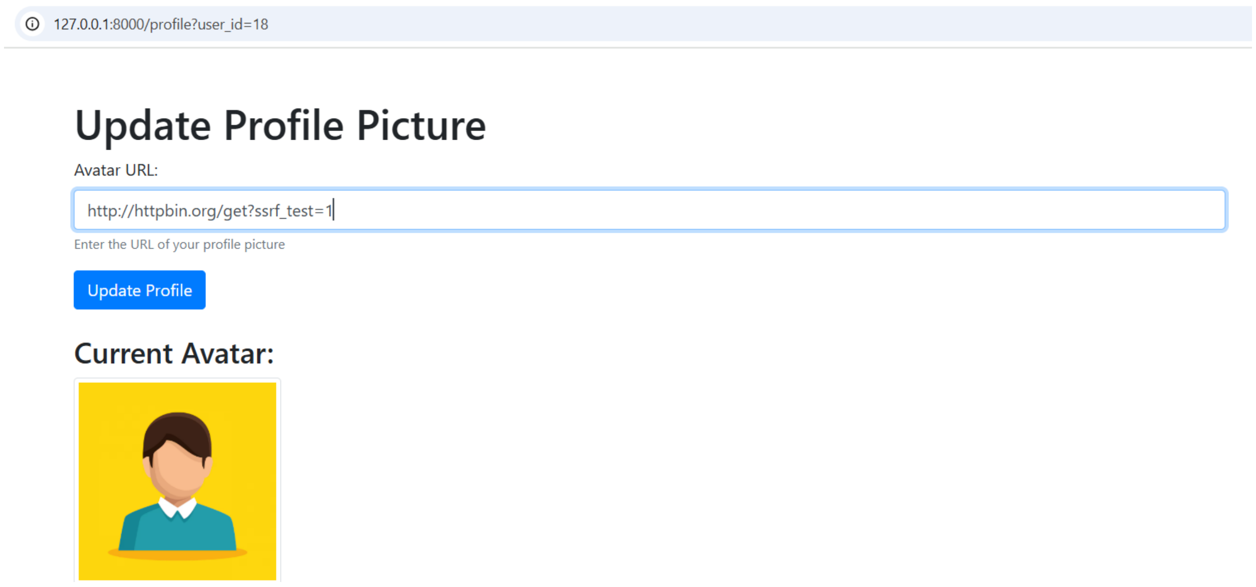
1. Basic SSRF to Internal Service

Payload: http://localhost/admin

Steps:

1. Go to ‘Edit profile’ in the dashboard
2. Enter ‘http://httpbin.org/get?ssrf_test=1’ as the avatar URL
3. Submit the form
4. The server attempts to fetch the internal admin page

Image of benign request:

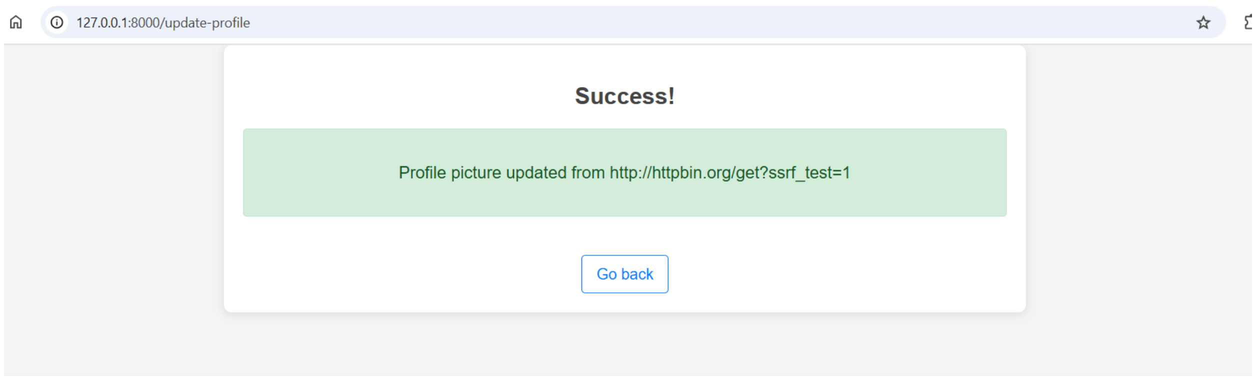


127.0.0.1:8000/profile?user_id=18

Update Profile Picture

Avatar URL: http://httpbin.org/get?ssrf_test=1

Enter the URL of your profile picture


127.0.0.1:8000/update-profile

Success!

Profile picture updated from http://httpbin.org/get?ssrf_test=1