Chapter 2: Application Security

Topic: Conduct Secure Code Review

Step 1: Set Up the Development Environment

```
┌──(kali㉿vbox)-[~]
└─$ python3 --version

Python 3.12.8

┌──(kali㉿vbox)-[~]
└─$ sudo apt update

[sudo] password for kali:
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [20.3 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [48.7 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [111 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [258 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [195 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [877 kB]
Get:8 http://kali.download/kali kali-rolling/non-free-firmware amd64 Packages [10.6 kB]
Get:9 http://kali.download/kali kali-rolling/non-free-firmware amd64 Contents (deb) [23.3 kB]
Fetched 70.5 MB in 41s (1704 kB/s)
2539 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(kali㉿vbox)-[~]
└─$ sudo apt install python3-venv

The following packages were automatically installed and are no longer required:
  libnghttp3-3              libqt6sql6t64      python3-lib2to3
  libpython3.11-dev         libqt6test6t64     python3-talloc
  libqt6dbus6t64            libqt6widgets6t64  python3-tdb
  libqt6gui6t64             libqt6xml6t64      python3.11
  libqt6network6t64         libwireshark17t64  python3.11-dev
  libqt6opengl6t64          libwiretap14t64    python3.11-minimal
  libqt6openglwidgets6t64   libwsutil15t64     tdb-tools
  libqt6printsupport6t64    python3-gpg
Use 'sudo apt autoremove' to remove them.

Installing:
  python3-venv

Installing dependencies:
  python3.12-venv

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 2539
  Download size: 7012 B
  Space needed: 34.8 kB / 7091 MB available
```

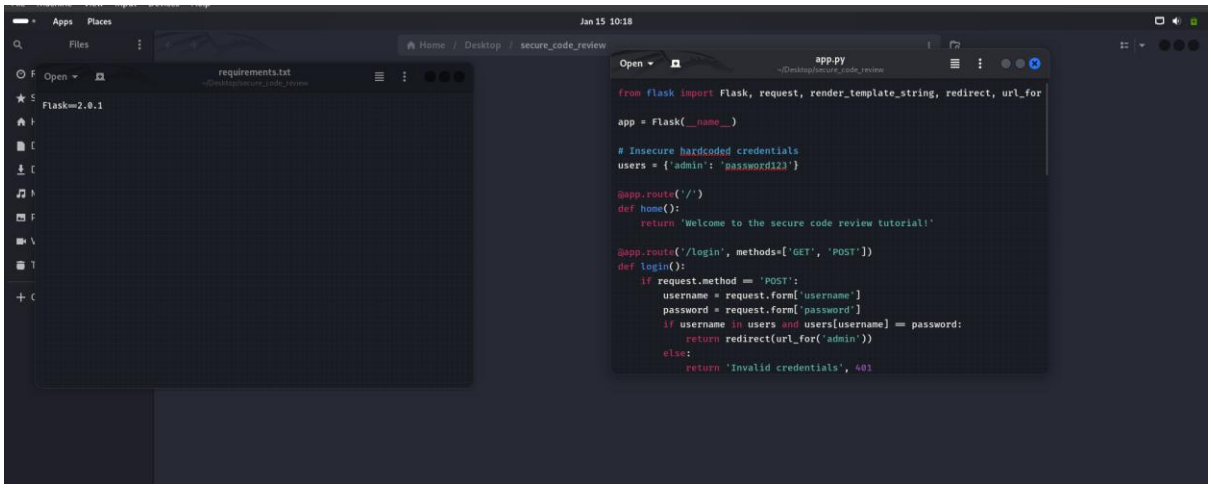
```
(venv)kali@vbox: ~/Desktop/secure_code_review
┌──(kali㉿vbox)-[~]
└─$ cd Desktop

┌──(kali㉿vbox)-[~/Desktop]
└─$ mkdir secure_code_review

┌──(kali㉿vbox)-[~/Desktop]
└─$ cd secure_code_review

┌──(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ python3 -m venv venv

┌──(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ source venv/bin/activate

┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ pip install Flask

Collecting Flask
  Downloading flask-3.1.0-py3-none-any.whl.metadata (2.7 kB)
Collecting Werkzeug>=3.1 (from Flask)
  Downloading werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Collecting Jinja2>=3.1.2 (from Flask)
  Downloading jinja2-3.1.5-py3-none-any.whl.metadata (2.6 kB)
Collecting itsdangerous>=2.2 (from Flask)
```

Step 2: Download the Sample Application

1. Create the Project Directory

```
┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ touch app.py

┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ touch requirements.txt

┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$
```

2. Set Up Files:

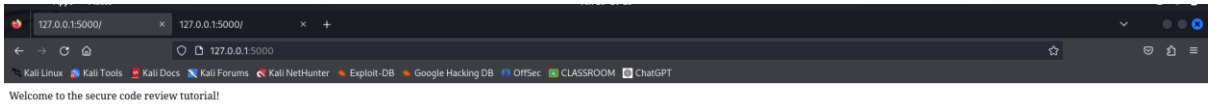


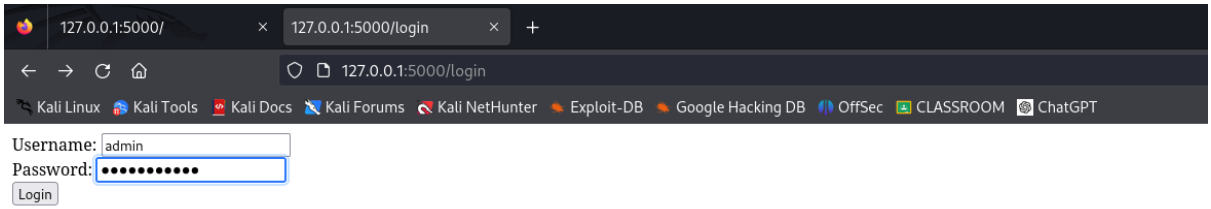
3. Install Dependencies:

```
┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ pip install -r requirements.txt

Collecting Flask==2.0.1 (from -r requirements.txt (line 1))
  Downloading Flask-2.0.1-py3-none-any.whl.metadata (3.8 kB)
Requirement already satisfied: Werkzeug>=2.0 in ./venv/lib/python3.12/site-packages (from Flask==2.0.1->-r requirements.txt (line 1)) (3
.1.3)
Requirement already satisfied: Jinja2>=3.0 in ./venv/lib/python3.12/site-packages (from Flask==2.0.1->-r requirements.txt (line 1)) (3.1
.5)
Requirement already satisfied: itsdangerous>=2.0 in ./venv/lib/python3.12/site-packages (from Flask==2.0.1->-r requirements.txt (line 1)
) (2.2.0)
Requirement already satisfied: click>=7.1.2 in ./venv/lib/python3.12/site-packages (from Flask==2.0.1->-r requirements.txt (line 1)) (8.
1.8)
Requirement already satisfied: MarkupSafe>=2.0 in ./venv/lib/python3.12/site-packages (from Jinja2>=3.0->Flask==2.0.1->-r requirements.t
xt (line 1)) (3.0.2)
Downloading Flask-2.0.1-py3-none-any.whl (94 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.8/94.8 kB 205.4 kB/s eta 0:00:00
Installing collected packages: Flask
  Attempting uninstall: Flask
    Found existing installation: Flask 3.1.0
    Uninstalling Flask-3.1.0:
      Successfully uninstalled Flask-3.1.0
Successfully installed Flask-2.0.1

┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ S
```

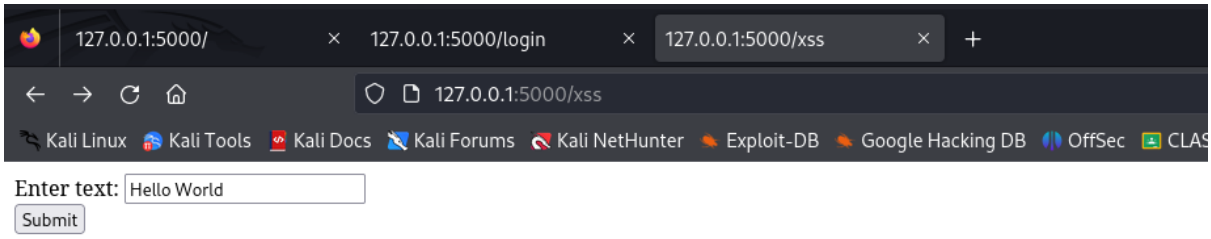
4. Run the Application:

```
┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ pip install werkzeug==2.0.3

Collecting werkzeug==2.0.3
  Downloading Werkzeug-2.0.3-py3-none-any.whl.metadata (4.5 kB)
Downloading Werkzeug-2.0.3-py3-none-any.whl (289 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 289.2/289.2 kB 215.8 kB/s eta 0:00:00
Installing collected packages: werkzeug
  Attempting uninstall: werkzeug
    Found existing installation: Werkzeug 3.1.3
    Uninstalling Werkzeug-3.1.3:
      Successfully uninstalled Werkzeug-3.1.3
Successfully installed werkzeug-2.0.3

┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ python app.py

 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 756-171-938
```





Where password is password123



```
┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ python app.py

 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 756-171-938
127.0.0.1 - - [15/Jan/2025 10:23:14] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jan/2025 10:23:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jan/2025 10:23:35] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [15/Jan/2025 10:25:01] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [15/Jan/2025 10:25:23] "POST /login HTTP/1.1" 302 -
127.0.0.1 - - [15/Jan/2025 10:25:23] "GET /admin HTTP/1.1" 200 -
127.0.0.1 - - [15/Jan/2025 10:26:24] "GET /xss HTTP/1.1" 200 -
127.0.0.1 - - [15/Jan/2025 10:26:38] "POST /xss HTTP/1.1" 200 -
127.0.0.1 - - [15/Jan/2025 10:26:48] "GET /xss HTTP/1.1" 200 -
```

Step 3: Review the Code

```
┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ pip install bandit

Collecting bandit
  Downloading bandit-1.8.2-py3-none-any.whl.metadata (7.0 kB)
Collecting PyYAML>=5.3.1 (from bandit)
  Downloading PyYAML-6.0.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (2.1 kB)
Collecting stevedore>=1.20.0 (from bandit)
  Downloading stevedore-5.4.0-py3-none-any.whl.metadata (2.3 kB)
Collecting rich (from bandit)
  Downloading rich-13.9.4-py3-none-any.whl.metadata (18 kB)
Collecting pbr>=2.0.0 (from stevedore>=1.20.0->bandit)
  Downloading pbr-6.1.0-py2.py3-none-any.whl.metadata (3.4 kB)
```

Bandit is a security-focused static analysis tool for Python code. It is used to identify potential security vulnerabilities in Python applications by scanning the code for common issues, such as:

1. **Hardcoded passwords** and credentials.
2. **Insecure usage of functions** that could be exploited, such as eval().
3. **Improper error handling** that could expose sensitive information.

4. **Insecure libraries** or modules that are known to be vulnerable.

5. **Weak cryptography or weak hashing algorithms**.

```
Successfully installed PyYAML-6.0.2 bandit-1.8.2 markdown-it-py-3.0.0 mdurl-0.1.2 pbr-6.1.0 pygments-2.19.1 rich-13.9.4 stevedore-5.4.0

┌──(venv)─(kali㉿vbox)-[~/Desktop/secure_code_review]
└─$ bandit -r .

[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 3.12.8
Working... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  1% 0:02:08[manager]       WARNING Test in comment: _lines is not a test name or id, ignoring
[manager]       WARNING Test in comment: is is not a test name or id, ignoring
[manager]       WARNING Test in comment: a is not a test name or id, ignoring
[manager]       WARNING Test in comment: dict is not a test name or id, ignoring
[manager]       WARNING Test in comment: of is not a test name or id, ignoring
[manager]       WARNING Test in comment: line is not a test name or id, ignoring
[manager]       WARNING Test in comment: number is not a test name or id, ignoring
[manager]       WARNING Test in comment: set is not a test name or id, ignoring
[manager]       WARNING Test in comment: of is not a test name or id, ignoring
[manager]       WARNING Test in comment: tests is not a test name or id, ignoring
[manager]       WARNING Test in comment: to is not a test name or id, ignoring
[manager]       WARNING Test in comment: ignore is not a test name or id, ignoring
Working... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  2% 0:00:37[manager]       WARNING Test in comment: tkelsey is not a test name or id, ignoring
Working... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  2% 0:00:36[manager]       WARNING Test in comment: catching is not a test name or id, ignoring
[manager]       WARNING Test in comment: expected is not a test name or id, ignoring
[manager]       WARNING Test in comment: exception is not a test name or id, ignoring
Working... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━  4% 0:00:25[tester]        WARNING nosec encountered (B108), but no failed test on line 62
Working... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:53
Run started:2025-01-15 15:30:41.951473
```

```
Code scanned:
        Total lines of code: 308633
        Total lines skipped (#nosec): 0

Run metrics:
        Total issues (by severity):
                Undefined: 0
                Low: 590
                Medium: 31
                High: 27
        Total issues (by confidence):
                Undefined: 0
                Low: 1
                Medium: 25
                High: 622
Files skipped (0):
```

The output from **Bandit** gives you an overview of the security issues found in your Python code, categorized by **severity** and **confidence**. Here's what each part of the result means:

1. **Total lines of code**:
    - The total number of lines of code Bandit scanned in your project, which is 308,633.
2. **Total issues (by severity)**:
    - This shows the number of issues Bandit found, categorized by how serious they are:
        - **Low**: 590 issues found that are of low severity. These are issues that could potentially lead to problems but aren't immediately dangerous.
        - **Medium**: 31 issues with medium severity.
        - **High**: 27 issues with high severity, meaning these are likely to have a serious impact on security.
3. **Total issues (by confidence)**:
    - Bandit assigns a confidence level based on how sure it is about the detected issue:
        - **Low confidence**: 1 issue where Bandit is not very sure about the vulnerability.
        - **Medium confidence**: 25 issues where Bandit has a moderate level of confidence.
        - **High confidence**: 622 issues where Bandit is highly confident there's a potential security issue.
4. **Files skipped (0)**:
    - Bandit didn't skip any files, meaning all files in the project were scanned.

Step 4: Fix Vulnerabilities

Hardcoded credentials refer to situations where sensitive information, like passwords, is stored directly in the source code. This is risky because attackers could easily view this information if they gain access to the codebase.

```
from flask import Flask, request, render_template_string, redirect, url_for

app = Flask(__name__)

# Insecure hardcoded credentials
users = {'admin': 'password123'}
```

What's wrong: This line stores the password password123 in plaintext.

Why it's a problem: Anyone who gains access to the codebase can view these credentials, and if the code is pushed to a public repository (like GitHub), anyone can easily access the admin account.

Replace Hardcoded Credentials:

```
from flask import Flask, request, render_template_string, redirect, url_for
from werkzeug.security import generate_password_hash, check_password_hash
from flask import escape

app = Flask(__name__)

# Store hashed passwords
users = {'admin': generate_password_hash('password123')}
```

Password: password122

Invalid credentials

Password: password123

Welcome to the admin page!

Conclusion:

In this tutorial, you learned how to identify and secure common vulnerabilities in a Flask web application. The key focus was on:

- Preventing Cross-Site Scripting (XSS) attacks by encoding user input to treat it as plain text.
- Implementing CSRF (Cross-Site Request Forgery) protection using secure tokens to ensure that form submissions are intentional.
- Protecting sensitive data by avoiding hardcoding secrets and ensuring passwords are encrypted using hashing.

These steps are fundamental to ensuring web applications are secure, protecting both users and systems from common types of attacks.

Main Keywords You Learned:

1. **XSS (Cross-Site Scripting)**
2. **CSRF (Cross-Site Request Forgery)**
3. **Sensitive Data Exposure**
4. **Flask-WTF (CSRF Token Protection)**
5. **Password Hashing (bcrypt)**
6. **Escape Function (for XSS Prevention)**
7. **Secure Input Validation**
8. **Flask Framework**

These keywords are central to improving the security of web applications and preventing common security issues.