🔐 Secure Coding Practices in Web Applications

Focus Areas: XSS, CSRF, SQL Injection

Web applications are exposed to various security threats due to improper coding practices. Among the most common and severe vulnerabilities are Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and SQL Injection. This document outlines each threat and highlights secure coding techniques to prevent them.

2. 🛡️ Cross-Site Request Forgery (CSRF)

🔍 What is CSRF?

CSRF exploits the trust that a web application has in the user's browser. An attacker tricks the user into submitting unwanted actions (like changing their email) on an authenticated session.

🧠 Attack Vector:

A malicious website submits a form or makes a GET/POST request to another site where the user is already logged in.

✅ Secure Coding Practices:

- **Anti-CSRF Tokens:** Generate a unique token for each session/form and validate it on submission.
- **SameSite Cookies:** Use `SameSite=Lax` or `Strict` to restrict cross-site cookies.
- **Double Submit Cookies:** Set a cookie and include the same value in a form/header for verification.

✅ Example in Flask-WTF:

```
from flask_wtf import FlaskForm
from wtforms import StringField
from wtforms.validators import DataRequired

class ProfileForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired()])
```

```
<form method="post">
  {{ form.csrf_token }}
  {{ form.email.label }} {{ form.email() }}
</form>
```

🔐 General Secure Coding Practices

Practice	Description
✅ Input Validation	Whitelist approach – only allow expected formats
✅ Output Encoding	Encode data based on context (HTML, URL, JavaScript)
✅ Least Privilege	DB accounts should have minimal privileges
✅ Secure Headers	Use headers like `X-Frame-Options`, `X-XSS-Protection`, `Content-Security-Policy`
✅ Session Management	Use secure, HTTP-only cookies; regenerate session IDs
✅ Logging and Monitoring	Log access attempts, unusual activities, and validation failures

🧩 Summary Table

Threat	Prevention
XSS	Escape output, CSP, input validation
CSRF	CSRF tokens, SameSite cookies
SQLi	Prepared statements, input validation

📚 References

- OWASP Top 10
- OWASP Cheat Sheet Series
- Mozilla CSP Guide
- Flask Security Docs