Project Proposal: Password-less Authentication System using OTPs

Contents

1. Title of the Project

Password-less Authentication System using One-Time Passwords (OTPs)

---

2. Team Information

Name: Uzair Zahid	**ID:** 22i-1619	**Role:** Project Lead, Documentation
Name: Haziq Hussain	**ID:** 22i-1685	**Role:** Frontend Developer
Name: Hussain Abdullah	**ID:** 22i-1639	**Role:** Backend Developer
Name: Hassaan Siddiqui	**ID:** 22i-1642	**Role:** Tester

---

3. Problem Statement

Traditional password-based authentication is prone to phishing, brute force, credential stuffing, and password reuse attacks. Many users tend to use weak passwords or the same password for multiple accounts, which adds to the vulnerability of their accounts to being compromised.

A **password-less authentication** system using OTPs helps solve all these issues by removing the need for static passwords. Each login is authenticated with a short-lived, one-time password delivered over a secure channel (e.g., email or SMS), so stolen credentials are useless after the first use.

---

4. Objectives of the Project

- Develop a web application that implements password-less login using OTPs.
- Enhance user authentication security by eliminating static passwords.
- Demonstrate secure OTP generation, delivery, and validation mechanisms.
- Integrate secure coding practices and apply security principles throughout the Secure SDLC.
- Deliverables: threat model, secure SDLC activities, test report, demo presentation.

---

5. Proposed Solution

- **System Overview:**

  The web app will allow users to log in without a password. Instead, users will enter their username/email, receive a one-time code via email (or SMS), and use that code to authenticate.

- **Security Principles:**
  - OTPs will be short-lived (e.g., 3 minutes validity).
  - Codes will be randomly generated using secure libraries.
  - Rate limiting will prevent brute force attempts.
  - OTPs will not be reused.

- All communications will occur over HTTPS.

- **Threats & Countermeasures:**
  - **Threat:** OTP interception (via insecure channel).
    Countermeasure: Enforce TLS encryption and use email delivery initially (extendable to SMS/app).
  - **Threat:** Brute forcing OTPs.
    Countermeasure: Limit attempts and lock accounts temporarily after failures.
  - **Threat:** Replay attacks.
    Countermeasure: Expire OTPs immediately after use.
  - **Threat:** Session hijacking.
    Countermeasure: Use secure cookies and session tokens.

---

6. Methodology (Secure SDLC Integration)

1. **Requirements Phase:**
   - Define functional requirements (OTP login, user registration).
   - Define security requirements (OTP validity, transport encryption, audit logging).
2. **Design Phase:**
   - Create architecture diagram (frontend + backend + database).
   - Identify threats using STRIDE.
   - Plan countermeasures (e.g., input validation, secure session management).
3. **Implementation Phase:**
   - Use secure coding standards (e.g., OWASP Top 10 compliance).
   - Implement OTP generation with cryptographically secure libraries.
   - Use parameterized queries to prevent SQL injection.
4. **Testing Phase:**

    - Perform unit and integration testing.
    - Conduct penetration testing focusing on OTP logic, replay, and brute force.
    - Verify compliance with OWASP guidelines.
5. **Deployment Phase:**
    - Deploy on a secure web server with HTTPS.
    - Enforce secure session management.
    - Maintain audit logs for authentication events.
6. **Maintenance Phase:**
    - Regularly update libraries and dependencies.
    - Monitor for suspicious login attempts.
    - Plan for scalability (e.g., SMS integration, push notifications).

---

7. Tools and Technologies

- **Languages/Frameworks:** Python (Flask/Django) or JavaScript (Node.js/Express) for backend, HTML/CSS/JS for frontend.
- **Database:** MySQL or PostgreSQL (storing user info and OTP logs securely).
- **Security Libraries:**
    - Python: secrets or PyOTP for OTP generation.
    - Node.js: otplib for OTPs.
- **Testing:** OWASP ZAP for vulnerability scanning.
- **Other Tools:** GitHub for version control.

---

8. Expected Deliverables

- A working prototype of a web app with password-less OTP login.
- Documentation (system design, threat model, and secure SDLC integration).
- Final project report.
- Project presentation/demonstration.

---

9. Timeline

Week	Milestone
1	Requirement gathering & literature review
2	System design & threat modeling
3–4	Backend & OTP implementation
5	Frontend integration
6	Security testing & fixes
7	Documentation & final testing
8	Submission & presentation

---

10. References

- OWASP Authentication Cheat Sheet
- NIST SP 800-63B: Digital Identity Guidelines
- PyOTP / otplib documentation
- Course textbook & lecture notes