Security Policies and Guidelines for Booking System

1. Introduction

This document outlines security policies for the booking system to ensure data integrity, confidentiality, and availability. The goal is to implement baseline security practices and establish an approach to maintaining security within the system's development, deployment, and operational phases.

2. Relevant Standards and Guidelines

- **OWASP Top 10**: This document reflects security practices that address common vulnerabilities, including Injection, Authentication Failures, and Security Misconfigurations.
- **ISO/IEC 27001**: While this project doesn't formally align with ISO 27001, certain principles, such as access control, incident response, and regular monitoring, are implemented.
- **ISO/IEC 19249**: Security Design Patterns from ISO/IEC 19249 have been referenced, particularly regarding secure data handling and access control practices.

3. Secure Coding Practices

Secure coding practices are applied throughout the booking system's codebase to mitigate potential security threats, particularly those identified in the OWASP Top 10.

- **Input Validation and Sanitization**: User inputs are validated and sanitized to prevent Injection attacks. This includes sanitizing inputs for any user-generated content, query parameters, and form data submissions.
- **Output Encoding**: Data displayed to users, such as booking details, is properly encoded to avoid Cross-Site Scripting (XSS) attacks.
- **Error Handling and Logging**: Error messages are generalized to prevent information disclosure. Sensitive information is not exposed in logs, and server-side errors are logged securely.
- **Session Management**: Session tokens are securely stored, and session expiration is enforced to reduce exposure to session hijacking.

4. Authentication and Password Policies

The system implements secure authentication mechanisms to prevent unauthorized access.

- **Password Hashing**: Passwords are stored as salted hashes using bcrypt. Alternatives like Argon2 and PBKDF2 are also recommended for future scalability.
- **Multi-Factor Authentication (MFA)**: For administrative accounts, MFA is recommended (implemented via Speakeasy in this system) to add an extra layer of security.
- **Authentication Tokens**: JWT tokens are used for user sessions. Tokens are stored securely and periodically rotated to minimize exposure risk.
- **Access to Source Code**: The source code is hosted on GitHub, with access restricted via SSH keys. Only authorized individuals (in this case, the developer) can access the repository.

5. Access Control Policies

Access control mechanisms prevent unauthorized actions within the system.

- **Role-Based Access Control (RBAC)**: Permissions are assigned based on user roles (e.g., user vs. admin). Admins have access to additional controls, such as managing rooms and bookings.
- **Principle of Least Privilege**: Users are only granted permissions necessary for their role, minimizing the risk of unauthorized access to sensitive functions.
- **Logging and Monitoring**: Access logs are maintained to track any administrative actions. Although this is a student project, basic logging can help trace unauthorized access if an incident occurs.

6. Incident Response Plan

The incident response plan outlines steps to be taken if a security incident is detected.

- **1. Preparation**:
  - Regularly back up the database and maintain access logs.
  - Monitor the system for unusual activities, such as multiple failed login attempts.
- **2. Identification**:
  - Use monitoring tools (e.g., application logs) to detect unusual behavior.
  - Be vigilant for alerts related to unauthorized access, errors, or other anomalies.
- **3. Containment**:
  - Restrict access to the system if a breach is suspected.
  - Temporarily disable user accounts if credentials are compromised.
- **4. Eradication**:
  - Identify and patch vulnerabilities that allowed the attack.
  - Remove any malicious code or backdoors from the system.
- **5. Recovery**:
  - Restore the system from a secure backup if necessary.
  - Notify affected users if personal information was exposed.
- **6. Lessons Learned**:
  - Document the incident, including root causes and remediation actions.
  - Update security policies and practices based on insights gained from the incident.

7. Secure Deployment and Access Controls

Secure deployment practices ensure that only authorized access and deployments occur.

- **SSH Access**: Access to the production environment is restricted through SSH, with access limited to the developer.
- **Environment Variables**: Sensitive information, such as API keys and database credentials, are stored in environment variables, and `.env` files are added to `.gitignore` to avoid exposure in source control.

8. Compliance Requirements

To meet basic compliance principles:

- **Data Privacy**: User data, such as booking information, is securely stored in the database and is accessible only to authorized roles.
- **Regular Audits**: While formal audits aren't feasible for a student project, the system should undergo periodic security reviews to identify and patch potential vulnerabilities.
- **Documentation**: Ensure all code changes, incident responses, and security patches are documented for future reference.

9. Summary of Key Security Practices

This document has outlined security policies for the booking system, focusing on:

- Adherence to OWASP Top 10 principles to mitigate common web vulnerabilities.
- Authentication practices, including password hashing and MFA.
- Access control policies that enforce the principle of least privilege.
- A basic incident response plan for security incidents.
- Secure coding practices and deployment controls.

Implementing and maintaining these security policies will contribute to a secure and reliable booking system, fostering user confidence and system integrity.

- End -