# Security risk assessment report

|  |
| --- |
| **Part 1: Select up to three hardening tools and methods to implement** |
| 1. Employees Share Passwords 2. Admin Password for Database is Set to Default 3. Firewalls Do Not Filter Traffic 4. Multifactor Authentication (MFA) is Not Used |
|

|  |
| --- |
| **Part 2: Explain your recommendations** |
| **1. Employees Share Passwords**   * **Risk:** Increases the likelihood of unauthorized access. * **Recommended Fix:** Implement NIST password policies to ensure:   + Unique, strong passwords per user.   + Password expiration and change policies.   + Employee education on password security.   **2. Admin Password for Database is Set to Default**   * **Risk:** Default passwords make the database highly vulnerable. * **Recommended Fix:**   + Change the default password immediately.   + Use a strong, randomly generated password.   + Restrict database access to authorized personnel.   **3. Firewalls Do Not Filter Traffic**   * **Risk:** Allows unrestricted inbound and outbound network traffic. * **Recommended Fix:** Implement an **Intrusion Prevention System (IPS)**:   + Configure firewalls with strict access control rules.   + Monitor traffic for anomalies and suspicious behavior.   + Block unauthorized IPs and malicious activities.   **4. Multifactor Authentication (MFA) is Not Used**   * **Risk:** Increases vulnerability to brute force and credential stuffing attacks. * **Recommended Fix:** Implement MFA using the following approach:   + Install necessary packages:   pip install pyotp qrcode pillow   * + Generate an MFA secret per user:   secret = pyotp.random\_base32()  db.execute("UPDATE users SET mfa\_secret = ? WHERE id = ?", secret, user\_id)   * + Validate MFA during login:   totp = pyotp.TOTP(user\_mfa\_secret)  if not totp.verify(request.form.get("mfa\_code")):  return apology("Invalid MFA Code", 403) |