# Manual Code Review Exercise Student Worksheet

**Name:** **\_\_\_\_** **Date:** **\_\_\_\_**

**Lab Partner:** **\_\_\_\_** **Section:** **\_\_\_\_**

## 🎯 Learning Objectives

By completing this worksheet, I will be able to:

* ☐ Understand what manual code review is and how it differs from automated testing
* ☐ Apply systematic code review methodology to identify security vulnerabilities
* ☐ Recognize common security anti-patterns through manual source code inspection
* ☐ Evaluate authentication, authorization, and input validation implementations
* ☐ Document security findings with clear explanations and remediation recommendations
* ☐ Understand the role of human judgment in security assessment

## 📋 Exercise 1: Application Discovery

### 1.1 Application Structure Analysis

# Navigate to the application directory  
cd /workspaces/Docker\_Sandbox\_Demo/samples/unsecure-pwa  
  
# List all files and directories  
ls -la

**Files found in the application:**

* ☐ main.py
* ☐ user\_management.py
* ☐ templates/ directory
* ☐ database\_files/ directory
* ☐ requirements.txt

### 1.2 Application Components Understanding

**Complete the table below by examining the files:**

| File/Directory | Purpose | Main Functions |
| --- | --- | --- |
| main.py | **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* |
| user\_management.py | **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* |
| templates/ | **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* |
| database\_files/ | **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* |

### 1.3 Initial Code Exploration

**Browse through the main application file:**

cat main.py

**Key observations about the application:**

1. What web framework is being used? **\_\_\_\_**
2. How many routes (web pages) does the application have? **\_\_\_\_**\_**\_\_\_\_**
3. What HTTP methods are supported? **\_\_\_\_**

## 📋 Exercise 2: Authentication Security Review

### 2.1 User Login Process Analysis

**Examine the user authentication process in user\_management.py:**

cat user\_management.py

**Find the retrieveUsers function and answer:**

1. **How are usernames checked against the database?**
2. **How are passwords verified?**
3. **What SQL query pattern is used for user lookup?**

* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### 2.2 Authentication Vulnerability Assessment

**🔍 Security Analysis Questions:**

1. **Is the SQL query construction secure? Why or why not?**
2. **What could happen if someone entered unusual characters in the username field?**
3. **Could an attacker bypass authentication? How?**

### 2.3 Password Security Review

**Examine how passwords are handled:**

1. **How are passwords stored in the database?**
   * ☐ Encrypted
   * ☐ Hashed
   * ☐ Plain text
   * ☐ Unknown/Need to investigate further
2. **What are the security implications of this password storage method?**

## 📋 Exercise 3: SQL Injection Vulnerability Hunt

### 3.1 Database Query Analysis

**Review all database queries in user\_management.py and identify vulnerable patterns:**

**Vulnerable Query #1:**

* **Location (function name):** **\_\_\_\_\_\*\*\_\_**\*\*
* **Line/Code:** **\_\_\_\_**\_\_\_\_\*\*
* **Why is it vulnerable?** **\_\_\_\_**\_\_\_\_\*\*
* **Attack example:** **\_\_\_\_**\_**\_\*\*\_\_**\*\*

**Vulnerable Query #2:**

* **Location (function name):** **\_\_\_\_\_\*\*\_\_**\*\*
* **Line/Code:** **\_\_\_\_**\_\_\_\_\*\*
* **Why is it vulnerable?** **\_\_\_\_**\_\_\_\_\*\*
* **Attack example:** **\_\_\_\_**\_**\_\*\*\_\_**\*\*

**Additional Vulnerable Queries Found:**

* **Query #3:** **\_\_\_\_\_\*\*\_\_**\*\*
* **Query #4:** **\_\_\_\_\_\*\*\_\_**\*\*

### 3.2 SQL Injection Impact Assessment

**For each vulnerable query, assess the potential impact:**

1. **Username lookup vulnerability could allow:**
   * ☐ Reading other users’ data
   * ☐ Modifying database records
   * ☐ Deleting user accounts
   * ☐ Administrative access
2. **Password check vulnerability could allow:**
   * ☐ Authentication bypass
   * ☐ Access to any user account
   * ☐ Administrative privileges
   * ☐ Data theft

### 3.3 SQL Injection Remediation

**How should these vulnerabilities be fixed?**

**Secure Query Pattern:**

# Instead of: cur.execute(f"SELECT \* FROM users WHERE username = '{username}'")  
# Use: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Why is this pattern more secure?**

## 📋 Exercise 4: Cross-Site Scripting (XSS) Analysis

### 4.1 Template Security Review

**Examine the HTML templates for XSS vulnerabilities:**

cat templates/index.html  
cat templates/signup.html  
cat templates/success.html

**XSS Vulnerability #1:**

* **Location (template file):** **\_\_\_\_**
* **Vulnerable code:** **\_\_\_\_**\_**\_\*\*\_\_**\*\*
* **Why is this dangerous?** **\_\_\_\_**

### 4.2 User Input Display Analysis

**Look at how user feedback is displayed:**

**Examine the listFeedback function in user\_management.py:**

**XSS Vulnerability #2:**

* **Location (function name):** **\_\_\_\_**
* **How user input is processed:** **\_\_\_\_\_\*\*\_\_**\*\*
* **Why is this vulnerable to XSS?** **\_\_\_\_\_\*\*\_\_**\*\*

### 4.3 XSS Attack Scenarios

**For each XSS vulnerability found, describe a potential attack:**

1. **Template XSS Attack Scenario:**
2. **Stored XSS Attack Scenario:**

## 📋 Exercise 5: Business Logic Security Review

### 5.1 Authentication Logic Analysis

**Review the complete authentication process:**

**🔍 Critical Thinking Questions:**

1. **Does the authentication logic check username AND password together?**
   * ☐ Yes, in a single query
   * ☐ No, they are checked separately
   * ☐ Unclear from the code
2. **What happens if an attacker uses different usernames for each check?**
3. **Is there a timing attack vulnerability in the authentication process?**

### 5.2 Access Control Review

**Examine route protection in main.py:**

1. **Are sensitive functions protected with authentication checks?**
   * ☐ Yes, all routes require authentication
   * ☐ No, some routes are unprotected
   * ☐ Mixed protection levels
2. **Can users access functionality they shouldn’t?** \*\*\*

### 5.3 Business Logic Vulnerabilities

**Identify any business logic flaws:**

**Logic Flaw #1:**

* **Description:** **\_\_\_\_\_\*\*\_\_**\*\*
* **Location:** **\_\_\_\_**\_\_\_\_\*\*
* **Impact:** **\_\_\_\_**\_\_\_\_\*\*

**Logic Flaw #2:**

* **Description:** **\_\_\_\_\_\*\*\_\_**\*\*
* **Location:** **\_\_\_\_**\_\_\_\_\*\*
* **Impact:** **\_\_\_\_**\_\_\_\_\*\*

## 📋 Exercise 6: Error Handling and Information Disclosure

### 6.1 Error Message Analysis

**Look for information disclosure vulnerabilities:**

1. **What happens when database operations fail?**
2. **Are detailed error messages shown to users?**
3. **Could error messages help an attacker?** \*\*\*

### 6.2 Debug Information Review

**Check application configuration:**

1. **Is debug mode enabled?**
   * ☐ Yes
   * ☐ No
   * ☐ Unknown
2. **What information might be disclosed in debug mode?** \*\*\*

## 📋 Exercise 7: Comprehensive Vulnerability Documentation

### 7.1 Vulnerability Summary Table

**Complete the table for all vulnerabilities found:**

| Vulnerability Type | Location | Risk Level | Description | Remediation |
| --- | --- | --- | --- | --- |
| SQL Injection | **\_\_\_\_**\_\_\_\_\*\* | Critical | **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* |
| XSS | **\_\_\_\_**\_\_\_\_\*\* | High | **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* |
| Auth Bypass | **\_\_\_\_**\_\_\_\_\*\* | High | **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* |
| **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* | **\_\_** | **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* |
| **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* | **\_\_** | **\_\_\_\_**\_\_\_\_\*\* | **\_\_\_\_**\_\_\_\_\*\* |

### 7.2 Risk Prioritization

**Rank the vulnerabilities by priority (1 = most critical):**

1. **Priority 1:** **\_\_\_\_\_\*\*\_\_**\*\* **Justification:** **\_\_\_\_\_\*\*\_\_**\*\*
2. **Priority 2:** **\_\_\_\_\_\*\*\_\_**\*\* **Justification:** **\_\_\_\_\_\*\*\_\_**\*\*
3. **Priority 3:** **\_\_\_\_\_\*\*\_\_**\*\* **Justification:** **\_\_\_\_\_\*\*\_\_**\*\*

### 7.3 Executive Summary

**Write a brief executive summary (3-4 sentences) describing the overall security posture of the application:**

## 📋 Exercise 8: Professional Security Report

### 8.1 Detailed Vulnerability Report

**Choose your most critical vulnerability and document it professionally:**

**Vulnerability Title:** **\_\_\_\_\_\*\*\_\_**\*\*

**Severity:** **\_\_\_\_\_\*\*\_\_**\*\*

**Location:** **\_\_\_\_**\_**\_\*\*\_\_**\*\*

**Description:**

**Technical Details:**

**Proof of Concept:**

**Business Impact:**

**Recommended Fix:**

**Timeline for Fix:** **\_\_\_\_**\_\_\_\_\*\*

### 8.2 Remediation Roadmap

**Create a remediation plan with priorities:**

**Phase 1 (Immediate - 1 week):**

* ☐ \*\*\*
* ☐ \*\*\*

**Phase 2 (Short-term - 1 month):**

* ☐ \*\*\*
* ☐ \*\*\*

**Phase 3 (Long-term - 3 months):**

* ☐ \*\*\*
* ☐ \*\*\*

## 🎓 Learning Reflection

### Reflection Questions

1. **How did manual code review differ from what you expected?**
2. **What types of vulnerabilities were easiest to find? Most difficult?**
3. **How does manual review complement automated security testing?**
4. **What additional skills would help you be a better code reviewer?**
5. **How would you integrate code review into a development process?**

### Key Learning Outcomes

**Rate your confidence level (1-5, where 5 is very confident):**

* Understanding manual code review methodology: \_\_\_/5
* Identifying SQL injection vulnerabilities: \_\_\_/5
* Recognizing XSS vulnerabilities: \_\_\_/5
* Analyzing authentication logic: \_\_\_/5
* Documenting security findings professionally: \_\_\_/5
* Prioritizing security vulnerabilities by risk: \_\_\_/5

### Real-World Application

1. **In what scenarios would manual code review be most valuable?**
2. **How could manual review skills help in your future career?**
3. **What ethical considerations are important for code reviewers?**

## 📚 Additional Research (Optional)

### Extended Learning Activities

1. **Research one of the vulnerabilities you found:**
   * Find real-world examples of this vulnerability
   * Research recent news stories involving this vulnerability type
   * Look up the vulnerability in the OWASP Top 10
2. **Explore secure coding practices:**
   * Research secure coding guidelines for the programming language used
   * Find examples of secure implementations of the vulnerable patterns you identified
3. **Professional development:**
   * Research career paths that involve security code review
   * Look up security code review certifications or training programs

**📝 Instructor Use Only:**

* **Completion Time:** **\_** minutes
* **Assistance Required:** **\_\_\_\_**
* **Key Challenges:** **\_\_\_\_**\_\_\_\_\*\*
* **Suggestions for Improvement:** **\_\_\_\_**\_\_\_\_\*\*

**🔍 Great job completing the manual code review exercise! These skills will serve you well in understanding how security vulnerabilities are identified and remediated in real-world applications.**