



Centurion  
UNIVERSITY  
SCHOOL OF COMPUTER SCIENCE

School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

## **Applied and Action Learning** (Learning by Doing and Discovery)

**Name of the Experiment :** Audit 101 – Smart Contract Vulnerabilities

### \* Coding Phase: Pseudo Code / Flow Chart / Algorithm

Algorithm

1. **Start**
2. **Open Remix IDE** and create a new Solidity file (e.g., `VulnerabilityDemo.sol`).
3. **Write a smart contract** intentionally containing common vulnerabilities such as:
  - o Reentrancy
  - o Integer overflow/underflow
  - o Unchecked external call
  - o Missing access control
4. **Compile the contract** using the Solidity compiler.
5. **Deploy the contract** in Remix using a local environment (Remix VM).
6. **Perform transactions** to test the contract's functions and observe abnormal behaviors or exploitable outcomes.
7. **Analyze vulnerabilities** by identifying how they affect the contract logic or funds.
8. **Implement fixes** (e.g., using `ReentrancyGuard`, `require` checks, or `SafeMath`).
9. **Re-compile and re-test** to ensure vulnerabilities are resolved.
10. **Document the findings** including:
  - o Vulnerability type
  - o Cause
  - o Fix implemented
  - o Final audit status
11. **Stop**

### \* Softwares used

1. **Remix IDE** – For writing, deploying, and testing smart contracts.
2. **Solidity Compiler (Solc)** – To compile and check the smart contract code.
3. **MetaMask** – For interacting with the deployed contract using a Web3 wallet.
4. **Ganache** – For creating a local blockchain environment to simulate transactions.
5. **MythX / Slither (Optional)** – For automated smart contract vulnerability analysis and security auditing.

## \* Implementation Phase: Final Output (no error)

### ◆ Steps of Smart Contract Audit

1. **Code Review & Architecture Analysis**
    - Auditors study the smart contract's code structure, architecture, and logic flow.
    - Identify dependencies and potential integration issues.
  2. **Unit Testing**
    - Run multiple test cases for each function.
    - Ensure all functions behave as expected under various conditions.
  3. **Vulnerability Detection**
    - Use manual and automated tools (like Slither, MythX) to find bugs such as reentrancy, overflow, or access control flaws.
  4. **Initial Report Preparation**
    - Prepare a draft report listing all discovered issues, categorized by severity (high, medium, low).
    - Share with the development team for fixes.
  5. **Re-Audit and Final Report**
    - Verify that all reported issues are fixed.
    - Publish the final audit report with verified results and security recommendations.
- Types of Smart Contract Audit
    1. **Manual Audit**
      - Conducted by human experts reviewing the code line-by-line.
      - Detects logic flaws and complex vulnerabilities that tools might miss.
    2. **Automated Audit**
      - Uses specialized software to scan for known vulnerabilities.
      - Faster and cost-efficient, but may miss deeper logical issues.
    3. **Hybrid Audit (Manual + Automated)**
      - Combines both methods for maximum accuracy and coverage.
      - Considered the most effective approach for professional audits.



- Comprehensive Vulnerability Analysis:**

Performed detailed security checks on smart contracts to identify potential issues like reentrancy, overflow, and access control flaws through manual and automated reviews.

- Code Reliability & Risk Mitigation:**

Enhanced contract safety by validating logic, transaction flow, and data integrity to prevent exploits and ensure secure execution before deployment.

- Standardized Audit Framework:**

Applied Ethereum-based audit practices using tools like Slither and Mytril, ensuring consistency, transparency, and adherence to smart contract security standards.

- Scalability & Continuous Improvement:**

Established an adaptable audit setup supporting re-audits, modular analysis, and regular updates for future enhancements and learning applications.

## \* Observations

- Identified and analyzed key vulnerabilities in smart contracts like reentrancy and overflow errors.
- Understood the importance of security audits to ensure safe and reliable blockchain deployment.
- Gained practical insight into using audit tools and frameworks for code verification and risk prevention.

## ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
<b>Total</b>	<b>50</b>		

*Signature of the Student:*

Name :

Regn. No. :

Page No.....

*Signature of the Faculty:*