Centurion University
Shaping Lives... Empowering Communities...

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 Experiement : Audit 101 – Smart Contract Vulnerabilities

Objective/Aim:

To understand common vulnerabilities in smart contracts and learn how to detect and prevent them through auditing and best security practices.

Apparatus/Software 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.

Theory:

A **smart contract audit** is a systematic examination of the code that runs on the blockchain to identify security flaws, logical errors, and optimization issues.
Since smart contracts are **immutable** once deployed, auditing is critical to prevent financial losses or exploitation.

Common Smart Contract Vulnerabilities:

1. **Re-entrancy Attack**
   - Occurs when a function makes an external call to another untrusted contract before updating its state.
   - Example: The DAO hack (2016).
   - **Prevention:** Use checks-effects-interactions pattern and ReentrancyGuard.
2. **Integer Overflow / Underflow**
   - Happens when arithmetic operations exceed or go below the storage limit.
   - **Prevention:** Use SafeMath library (in older Solidity versions) or Solidity's built-in overflow checks (v0.8+).
3. **Unchecked External Calls**
   - External contract calls might fail silently.
   - **Prevention:** Always check the return value of low-level calls (call, delegatecall).
4. **Denial of Service (DoS)**
   - Attackers block contract functions by consuming all gas or locking logic flow.
   - **Prevention:** Avoid looping over unbounded data and handle errors properly.

Theory:

5. **Front Running**
   - Attackers monitor the mempool and submit transactions with higher gas fees to execute before others.
   - **Prevention:** Use commit-reveal schemes or private transactions.
6. **Access Control Issues**
   - Occur when privileged functions (like withdraw(), mint(), etc.) lack proper restrictions.
   - **Prevention:** Use onlyOwner modifiers and role-based access control.
7. **Timestamp Dependence**
   - Using block.timestamp for logic decisions can be manipulated slightly by miners.
   - **Prevention:** Avoid using timestamps for critical operations.

Procedure:

1. Open Remix IDE in a browser.
2. Create a new Solidity file, e.g., VulnerabilityDemo.sol.
3. Write a smart contract intentionally containing common vulnerabilities such as:
   - Reentrancy
   - Integer Overflow/Underflow
   - Unchecked External Call
   - Missing Access Control
4. Compile the contract using the Solidity Compiler.
5. Deploy the contract using a local environment (Remix VM).
6. Perform transactions and test functions to observe abnormal or exploitable behavior.
7. Identify vulnerabilities and analyze their causes and effects on contract logic and funds.
8. Implement security fixes (e.g., using ReentrancyGuard, require() checks, or SafeMath).
9. Re-compile and re-test to ensure vulnerabilities are fixed.
10. Document the findings including:
    - Vulnerability Type
    - Cause
    - Fix Implemented
    - Final Audit Status

Observation:

1 Identified and analyzed key vulnerabilities in smart contracts like reentrancy and overflow errors.

2. Understood the importance of security audits to ensure safe and reliable blockchain deployment.

3. 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		
Total	**50**		

Signature of the Student:

Name :

Signature of the Faculty:

Regn. No. :

**As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.