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 the most common smart contract vulnerabilities, how they occur, and how they can be prevented during development and auditing.

Apparatus/Software Used:

a. Laptop/PC
b. Remix IDE or Hardhat environment
c. MetaMask
d. Solidity Compiler
e. Blockchain test network (e.g., Sepolia)
f. Code analysis tools (Mythril, Slither, or Remix Security Analyzer)

Theory/Concept:

Smart contracts are self-executing programs deployed on blockchains. While they enable decentralization and automation, bugs in the contract can lead to permanent loss of funds.

Below are the key smart contract vulnerabilities (similar to "key properties" section in your file):

✅ 1. Reentrancy

- Occurs when an external contract repeatedly calls back into the vulnerable function before it finishes execution.
- Impact: Drains ETH / tokens.

✅ 2. Integer Overflow & Underflow

- Arithmetic operations exceed their limit (before Solidity 0.8).
- Impact: Attackers manipulate balances or calculations.

✅ 3. Access Control Issues

- Functions are not properly restricted (e.g., missing onlyOwner).
- Impact: Attackers gain admin-level permissions.

✅ 4. Unchecked External Calls

- Using call() or interacting with other contracts without validating results.
- Impact: Funds locked, unexpected behavior, reentrancy risk.

Procedure:

1. Open Remix IDE and load a Solidity smart contract.
2. Identify functions that:
- Transfer ETH or tokens
- Update state variables
- Interact with external contracts
3. Run a static analysis using Remix Security Analyzer.
4. Change parts of the smart contract code to test vulnerability behavior:
- Remove access modifiers (onlyOwner)
- Use an external call and observe warnings
- Run reentrancy test using attacker contract
5. Compare the contract's behavior before and after introducing vulnerabilities.
6. Record the findings and ensure best practices (checks-effects-interactions, SafeMath, proper access control).

Observation

- Smart contracts become vulnerable when external calls, arithmetic operations, or permissions are not carefully controlled.
- A small change (like missing a require or modifier) can lead to **severe exploitation**, similar to the "avalanche effect" seen in SHA-256.
- Proper auditing and testing significantly reduce vulnerability risks.

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 :

Regn. No.

Signature of the Faculty: