



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 Experiment :** Audit 101 – Smart Contract Vulnerabilities

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

#### Algorithm:

- 1.Start
- 2.Open Remix IDE in a browser.
- 3.Create a new Solidity file named VulnerableContract.sol.
- 4.Write a smart contract with intentional vulnerabilities.
- 5.Compile the contract and check for warnings or compiler errors.
- 6.Deploy the vulnerable contract using MetaMask on a test network.
- 7.Analyze its behavior by performing function calls that exploit the weakness.
- 8.Identify and record the cause of vulnerability.
- 9.Modify the contract to fix the issue and redeploy it.
- 10.Re-test the contract to ensure the vulnerability no longer exists.
- 11.End

### \* Software used

- 1.Remix IDE
- 2.Solidity
- 3.MetaMask
- 4.Test Network (Sepolia)

## \* Testing Phase: Compilation of Code (error detection)

Open Remix IDE. Create a new file VulnerableContract.sol. Write Vulnerable Contract Code

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract VulnerableBank {
5     mapping(address => uint256) public balances;
6
7     function deposit() public payable {  infinite gas
8         balances[msg.sender] += msg.value;
9     }
10
11     function withdraw(uint256 _amount) public {  infinite gas
12         require(balances[msg.sender] >= _amount, "Insufficient balance");
13         (bool sent, ) = msg.sender.call{value: _amount}("");
14         require(sent, "Failed to send Ether");
15         balances[msg.sender] -= _amount;
16     }
17 }
18
  
```

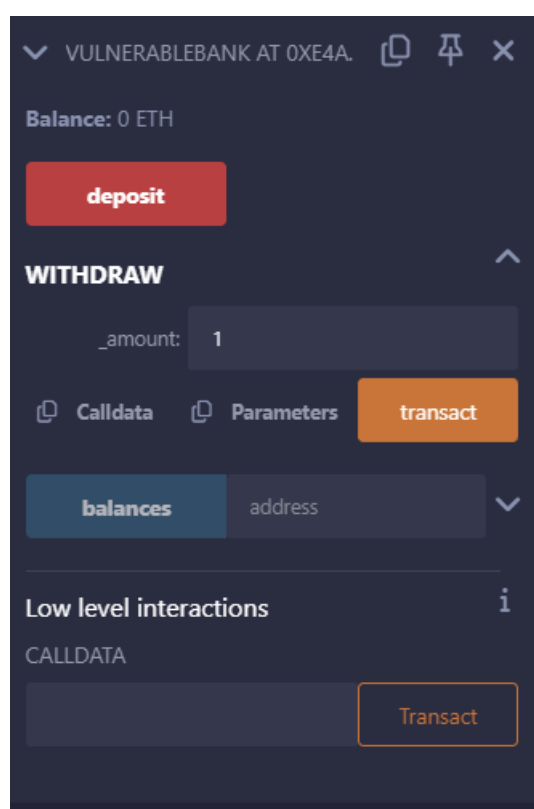
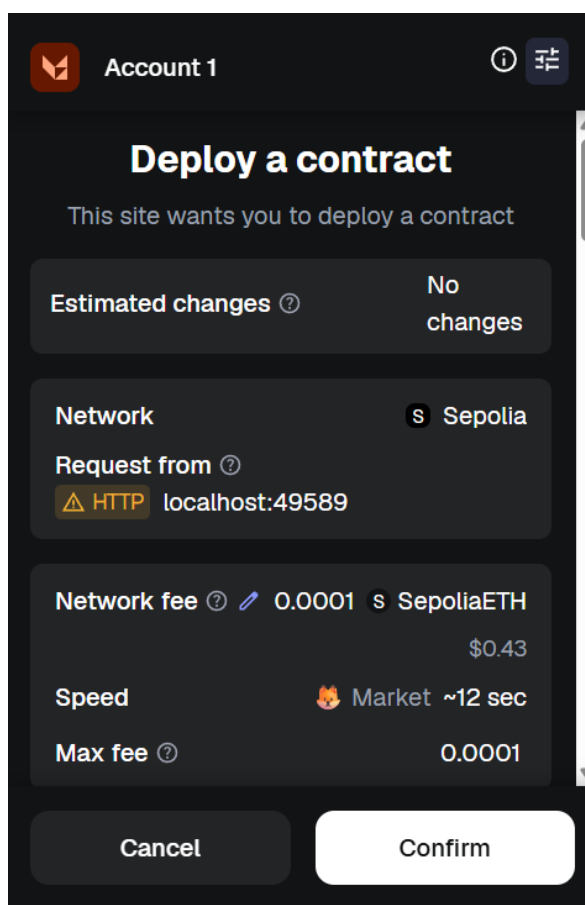
Deploy and Test:

Go to Deploy & Run Transactions.

Select Injected Provider - MetaMask and deploy the contract.

Deposit some ETH and attempt multiple withdrawals quickly.

Observe reentrancy behavior (if using test attacker contract).



0xE4a5139CE5b039b4B11a62b54a715b8108742E40

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

Applied and Action Learning

Successfully analyzed a vulnerable smart contract.  
Detected and mitigated a Reentrancy Attack vulnerability.  
The corrected contract follows best security practices.

```
creation of vulnerableBank pending...

view on Etherscan  view on Blockscout

[block:9546202 txIndex:4] from: 0xe4a...ca52e to: VulnerableBank.(constructor)
value: 0 wei data: 0x608...e0033 logs: 0 hash: 0x509...4e2b3 Debug
transact to VulnerableBank.withdraw pending ...

view on Etherscan  view on Blockscout

[block:9546214 txIndex:7] from: 0xe4a...ca52e
to: VulnerableBank.withdraw(uint256) 0xe4a...42e40 value: 0 wei data: 0x2e1...00001 Debug
logs: 0 hash: 0x027...73b74
```

## \* Observations

1. Identified and fixed a reentrancy vulnerability in the smart contract.
2. Understood the importance of auditing and secure coding before deployment.

## 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. :

**Signature of the Faculty:**

Page No.....

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