



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 : Security First – Understanding Blockchain Attacks

Objective/Aim:

To study and understand the major security threats and attack vectors in blockchain systems, their causes, and possible preventive mechanisms to ensure secure blockchain operation.

Apparatus/Software Used:

- ☐ Blockchain demo simulations (e.g., [Anders Brownworth Blockchain Demo](#))
- ☐ Remix IDE (for smart contract vulnerability demos)
- ☐ Test blockchain networks (Ethereum Testnet / Ganache)
- ☐ MetaMask Wallet

Theory/Concept:

Blockchain provides decentralized trust through cryptographic security, but it is still vulnerable to various **attacks** that exploit its design, consensus mechanism, or smart contract logic.

Common Types of Blockchain Attacks:

1. **51% Attack:**
 - Occurs when a single entity controls more than 50% of the network's hash power.
 - Allows attackers to reverse transactions and double-spend.
 - Example: Attack on Ethereum Classic in 2020.
2. **Sybil Attack:**
 - An attacker creates multiple fake nodes to influence consensus or network communication.
 - Prevented using proof-of-work or proof-of-stake mechanisms.
3. **Double-Spending Attack:**
 - The same cryptocurrency is spent twice by broadcasting two conflicting transactions.
 - Prevented through confirmations and consensus.
4. **Smart Contract Vulnerabilities:**
 - Poorly written smart contracts can be exploited.
 - Example: **The DAO Hack (2016)** due to re-entrancy bug.
5. **Phishing Attacks:**
 - Attackers trick users into revealing private keys or seed phrases using fake wallet sites.
6. **Routing and Eclipse Attacks:**
 - Target the P2P communication layer to isolate nodes and manipulate network data.

Procedure:

- ☐ Open [Anders Brownworth Blockchain Demo](#).
- ☐ Modify data in an existing block → observe how it invalidates subsequent blocks.
- ☐ Demonstrates **data tampering** and the **immutability** principle.
- ☐ Explore a **Proof-of-Work demo** and attempt to alter a mined block → notice difficulty in rematching the hash.
- ☐ Simulates the **computational cost of attacks**.
- ☐ Use **Remix IDE** → write a simple vulnerable smart contract:

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract Vulnerable {
5     mapping(address => uint256) public balances;
6
7     function deposit() external payable {
8         balances[msg.sender] += msg.value;
9     }
10
11     function withdraw() external {
12         uint256 bal = balances[msg.sender];
13         require(bal > 0, "No balance");
14
15         // vulnerable: external call before state update
16         (bool sent, ) = msg.sender.call{value: bal}("");
17         require(sent, "transfer failed");
18
19         balances[msg.sender] = 0;
20
21         // helper to check contract balance
22         function getBalance() external view returns (uint256) {
23             return address(this).balance;
24         }
25     }

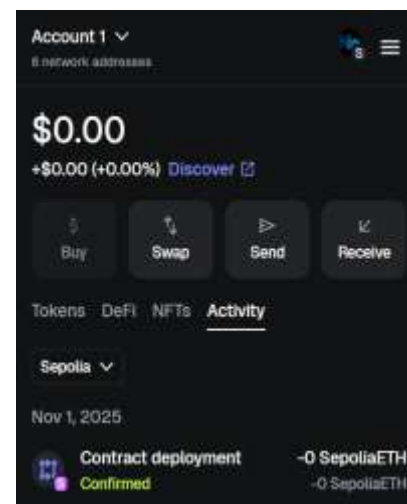
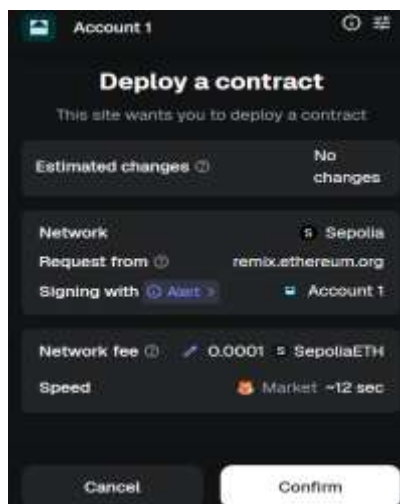
```

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 interface IVulnerable {
5     function deposit() external payable;
6     function withdraw() external;
7 }
8
9 contract Attacker {
10     IVulnerable public target;
11     address public owner;
12     uint256 public attackCount;
13
14     constructor(address _target) {
15         target = IVulnerable(_target);
16         owner = msg.sender;
17     }
18
19     // find attacker and deposit into target to setup victim balance
20     function fundedDeposit() external payable {
21         require(msg.value > 0, "Send ETH");
22         // deposit msg.value into victim contract under this contract's address
23         target.deposit{value: msg.value}();
24     }

```

- ☐ Deploy and test using two accounts on a testnet → simulate **re-entrancy attack**
- ☐ Discuss mitigations (use of ReentrancyGuard or updating state before transfer)



Observation Table:

Attack Type	Description / Simulation Result	Prevention Mechanism
Data Tampering	Changing block data broke hash linkage	Cryptographic hash & consensus verification
51% Attack	Hypothetical control over majority hash power	Decentralization, PoS mechanism
Smart Contract Bug	Vulnerable withdraw enabled re-entrancy simulation	Secure coding, auditing, reentrancy guard
Phishing Attack	Users tricked into fake MetaMask login	Verify URLs, never share seed phrase
Double-Spend (Concept)	Two conflicting transactions tested on testnet	Confirmations, strong consensus

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: