

# Lab Session: Security Audit

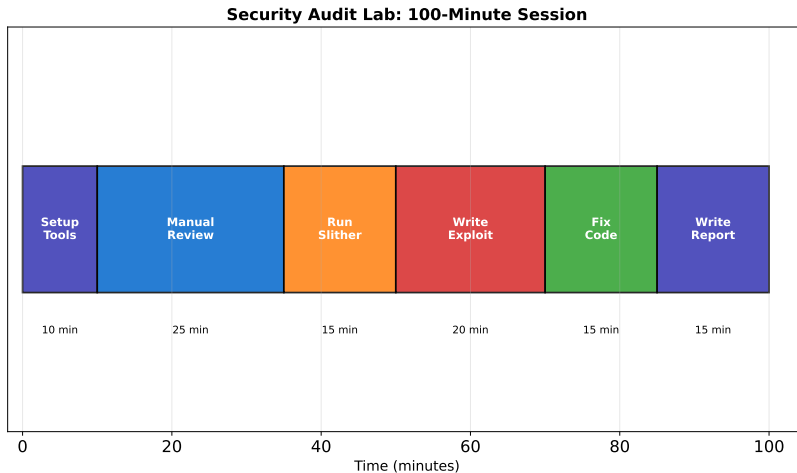
BSc Blockchain, Crypto Economy & NFTs

Course Instructor

Module F: Advanced Topics

By the end of this lab session, you will be able to:

- Perform manual security code review
- Use automated tools (Slither, Mythril)
- Write exploit contracts demonstrating vulnerabilities
- Implement secure fixes using best practices
- Create professional audit reports



*Complete security audit workflow from review to report*

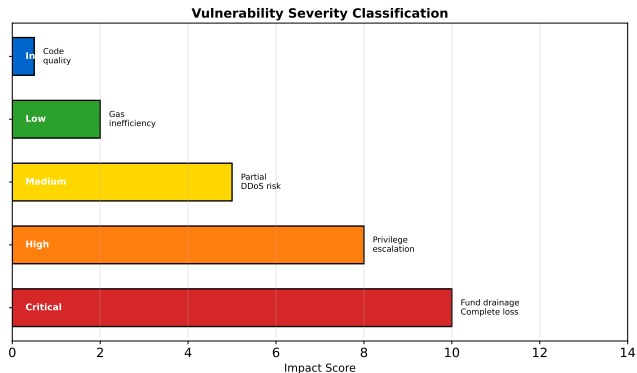
## Contracts to Audit:

- ❶ **VulnerableBank**: Reentrancy vulnerability
- ❷ **InsecureToken**: Integer overflow, access control
- ❸ **BadOracle**: Oracle manipulation

## Deliverables:

- Audit report with severity classifications
- Exploit contracts demonstrating vulnerabilities
- Secure implementations with fixes

# Vulnerability Severity Classification



*Critical and High severity issues require immediate fixes*

# Exercise 1: VulnerableBank (Reentrancy)

## Manual Review Tasks:

- 1 Read the withdraw function carefully
- 2 Identify external call before state update
- 3 Classify as Critical severity

## Fix with Checks-Effects-Interactions:

- Update balance BEFORE external call
- Or use OpenZeppelin ReentrancyGuard

## Exercise 2: InsecureToken (Multiple Issues)

### Vulnerabilities to Find:

- Integer overflow (Solidity < 0.8)
- Missing access control on mint
- Unprotected selfdestruct

### Fix:

- Upgrade to Solidity 0.8.0+
- Add onlyOwner modifier
- Initialize owner in constructor

## Exercise 3: BadOracle (Price Manipulation)

### Vulnerability:

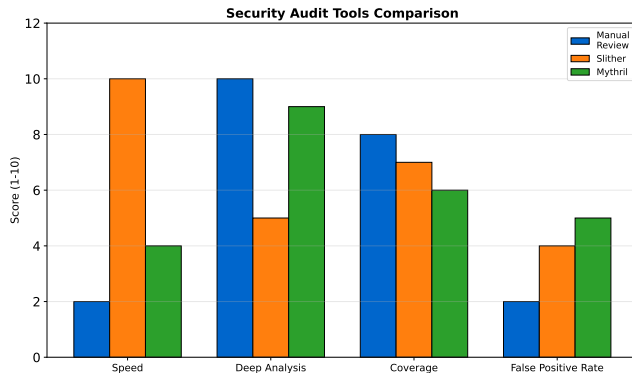
- Instant spot price from Uniswap
- Manipulable via flash loans

### Secure Alternatives:

- Uniswap V2 TWAP oracle (time-weighted)
- Chainlink price feeds (decentralized oracle)
- Hybrid approach with multiple sources



# Audit Tools Comparison



*Combine manual review with automated tools for best coverage*

## Slither (Static Analysis):

- Fast pattern-based detection
- Run: `slither contracts/VulnerableBank.sol`

## Mythril (Symbolic Execution):

- Deep path analysis
- Run: `myth analyze contracts/VulnerableBank.sol`

## Key Detectors:

- SWC-107: Reentrancy
- SWC-101: Integer overflow
- SWC-105: Unprotected withdrawal

## For Each Finding:

- ① **Title:** Descriptive vulnerability name
- ② **Severity:** Critical / High / Medium / Low
- ③ **Location:** Contract, function, line numbers
- ④ **Description:** What is the vulnerability
- ⑤ **Impact:** What can attacker achieve
- ⑥ **Recommendation:** How to fix

## Submit:

### ① Audit Report (PDF):

- Findings for all three contracts
- Severity classifications
- Fix recommendations with code

### ② Code Submissions:

- Exploit contracts (AttackBank.sol, etc.)
- Secure implementations
- Test suite proving fixes work

- Security audit combines manual review and automated tools
- Reentrancy is critical - use Checks-Effects-Interactions
- Solidity 0.8.0+ provides built-in overflow protection
- Never trust spot prices - use TWAP or Chainlink
- Exploit writing proves vulnerability severity
- Professional reports follow structured template