Smart Contract Vulnerabilities

This document contains smart contract vulnerabilities along with simplified definitions and concrete prevention strategies. It is structured for developers, auditors, and security engineers to use during audits, threat modeling, and protocol reviews.

1.Access Control Vulnerabilities

- **Definition**: Flaws in permission settings that allow unauthorized users to perform restricted actions.
- **Prevention:** Use Role-Based Access Control (RBAC), restrict function visibility, and regularly audit permissionsUnauthorized users can access restricted functions due to flawed permission checks.

2. Price Oracle Manipulation

- **Definition:** Manipulating asset prices via weak oracles to affect DeFi operationsprice feeds to distort DeFi calculations.
- **Prevention**: Use decentralized oracles (e.g., Chainlink), aggregate multiple datavalidate inputs, and use multiple sources.

3. Logic Errors (Business Logic Flaws)

- **Definition:** Incorrect contract logic leading to unintended outcomes like wrong payouts.
- **Prevention:** Perform unit testing, simulate edge cases, and get domain expert reviewsDefinition: Business logic flaws cause incorrect payouts or outcomes.

4. Lack of Input Validation

- **Definition**: Unchecked inputs may trigger undesired logic or security flaws.
- **Prevention**: Validate inputs, apply strict type checks, and use defensive programmingcoding.

5. Reentrancy Attacks

- **Definition:** Repeated external calls before state updates can drain funds.
- **Prevention**: Checks-Effects-Interactions pattern and reentrancy guards.

6. Unchecked External Calls

- **Definition:** External contract calls not verified for successcalls fail silently, leading to inconsistent states.
- **Prevention:** Always verify call success and handle errors gracefullyexceptions.

7. **Flash Loan Attacks**

- **Definition**: Exploiting uncollateralizedatomic loans to manipulate protocol logic.
- **Prevention**: Add time-locks, perform sanity checks, and monitor usagecontract behavior.

8. **Integer Overflow/Underflow**

- **Definition**: Arithmetic errors causing miscalculations or unauthorized mintingresult in balance miscalculations.
- **Prevention**: Use SafeMath or rely on built-in checks in Solidity >= 0.8.0’s built-in checks.

9. **Insecure Randomness**

- **Definition:** Predictable values compromise fairness in lotteriesoutcomes in games or airdrops.
- **Prevention**: Use Chainlink VRF or similar verifiable randomnessVRF (e.g., Chainlink VRF), avoid block variables.

10. Denial of Service (DoS)

- **Definition**: Resource exhaustion or forced reverts that block contract functionalityContract crashes or locks up due to resource exhaustion.
- **Prevention:** Optimize gas usage, limit untrusted loops, and add fail-safesuse and handle revert scenarios gracefully.

11. **Timestamp Dependency**

- **Definition**: Manipulable block.timestamp used for logic.
- **Prevention**: Prefer block.number and avoid critical logic based on timestampsMiners can manipulate block timestamps.

12. **Front-Running (Transaction Ordering Dependency)**

- **Definition**: Observing pending transactions and jumping the queue via higher gas.
- **Prevention**: Use commit-reveal schemes, Flashbots, or private transactionsand private mempools.

13. **DoS with Unexpected Revert**

- **Definition**: External calls revert unexpectedly, breaking execution flow.
- **Prevention**: Handle errors gracefully and include fallbackcause reverts that break functionality.

14. Gas Limit Vulnerabilities

- **Definition:** Insufficient gas allocation causes complex transactions to fail.
- **Prevention:** Estimate gas carefully and optimize code pathsContracts run out of gas due to complex logic.

15. **Ownership Vulnerabilities**

- **Definition:** Unauthorized users gaining or abusing contract ownership.
- **Prevention:** Secure ownership transfer, use Ownable, and enforce multi-sigtransfer of ownership privileges.

16. **Fallback Function Misuse**

- **Definition**: Exploiting fallback logic for unauthorized behavior.
- **Prevention:** Limit fallback logic, use receive and fallback properlyExploitable fallback or receive functions.

17. Insufficient Signature Verification

- **Definition:** Accepting invalid or replayed signatures.
- **Prevention:** Use ecrecover, nonces, timestamps, and EIP-712 structuresForged or reused signatures enable unauthorized actions.

18. Cross-Contract Execution Risks

- **Definition:** Vulnerabilities from dependent contracts.
- **Prevention:** Minimize inter-contract dependencies and use atomic operationsComplex contract interactions increase attack surface.

19. Uninitialized Storage Pointers

- **Definition:** Pointers left uninitialized can corrupt storage.
- **Prevention:** Always initialize variables and use static analysis toolsCan be used to overwrite critical storage.

20. Selfdestruct Vulnerabilities

- **Definition:** Contracts with open selfdestruct can be destroyed.
- **Prevention:** Lock down selfdestruct and require multi-sig for critical actionsdestroyed unexpectedly.

21. Gas Griefing

- **Definition:** Insufficient gas for subcalls causes partial transaction failure.
- **Prevention:** Let contracts set subcall gas limits rather than relying on user input.

22. Signature Replay Attacks

- **Definition**: Reusing a valid signature to trigger repeated unauthorized actions.
- **Prevention**: Implement nonce or timestamp-based validation.signature to perform duplicate actions.

23. Blockhash Dependence

- **Definition:** Relying on blockhash for randomness is insecure.
- **Prevention:** Avoid using blockhash, use verifiable random sources instead.

24. Time Manipulation Vulnerabilities

- **Definition:** Miners manipulate block.timestamp to skew logic.
- **Prevention:** Avoid timestamp-based logic for critical operations.

25. Event Emission Flaws

- **Definition:** Missing or incorrect events break tracking and transparency.
- **Prevention:** Emit accurate events on every state-changing action.

26. Improper Error Handling

- **Definition:** Misuse of require, revert, or assert can cause loss of funds.
- **Prevention:** Use each properly: require for validation, revert for conditional errors, assert for invariants.

27. Proxy Pattern Vulnerabilities

- **Definition:** Misconfigured proxies can be hijacked or misused.
- **Prevention:** Use audited proxy patterns (UUPS, Transparent), restrict upgrades.

28. Cross-Chain Vulnerabilities

- **Definition**: Attacks arising from multi-chain interactions.
- **Prevention:** Use verified bridges and validate logic on all involved chains.

29. Token Contract Vulnerabilities

- **Definition:** Bugs in token logic that affect supply or balances.
- **Prevention:** Follow standards like ERC20/721 strictly, and test token logic thoroughly.

30. Misconfigured Function Parameters

- **Definition:** Poorly defined parameters can lead to logic exploits.
- **Prevention:** Use strict type checks and validate inputsIncorrect parameters introduce bugs or exploits.

31. Event Emission Flaws

- **Definition:** Critical state changes not logged.
- **Prevention:** Emit events consistently for traceability.

32. Improper Error Handling

- **Definition:** Errors are swallowed or untracked.
- **Prevention:** Use require, assert, and revert correctly.

33. Improper Upgradeability

- **Definition:** Proxy contracts allow unauthorized logic replacement.
- **Prevention:** Control upgrade permissions and audit upgrade paths.

34. Cross-Chain Vulnerabilities

- **Definition:** Flaws in multi-chain interaction logic.
- **Prevention:** Use secure bridges, validate inputs from all chains.

35. Stale Price Usage

- **Definition:** Contracts use outdated oracle values.
- **Prevention:** Validate price freshness before relying on values.

36. Unlocked External Protocol Calls

- **Definition:** External integrations not authenticated.
- **Prevention:** Whitelist trusted contracts or protocols.

37. Deviation from Token Standard

- **Definition:** Incomplete or incorrect ERC20/ERC721 implementation.
- **Prevention:** Use OpenZeppelin or other verified implementations.

38. Event Emission Before Validation

- **Definition:** Logs occur before success conditions are met.
- **Prevention:** Emit events after all checks and state updates.

39. Logic Inversion in Conditionals

- **Definition:** Conditions accidentally reversed in checks.
- **Prevention:** Enforce test coverage on all condition branches.

40. Unsafe Batch Execution

- **Definition:** Inconsistent handling in batch operations.
- **Prevention:** Validate every element in looped operations.

41. Gas Bombs

- **Definition:** Malicious contracts designed to consume excess gas.
- **Prevention:** Isolate and cap external calls.

42. Token Whitelisting Flaws

- **Definition:** Allowing unauthorized tokens in interactions.
- **Prevention:** Restrict to audited/verified tokens only.

43. Unrestricted Withdrawals

- **Definition:** Lack of withdraw access controls.
- **Prevention:** Authenticate withdrawal rights.

44. Contract Size Limit Violations

- **Definition:** Contract exceeds EVM code size limit (24KB).
- **Prevention:** Refactor into modular components or proxies.

45. Merkle Proof Misuse

- **Definition:** Incorrect Merkle root validation leads to unauthorized claims.

- **Prevention:** Use audited libraries for Merkle proof verification.

46. Double Spend via Async Withdraw

- **Definition:** Funds withdrawn twice due to async logic.
- **Prevention:** Lock state before external calls.

47. Reward Inflation Bugs

- **Definition:** Rewards accumulate faster than intended.
- **Prevention:** Validate accrual math and upper limits.

48. Arbitrary Storage Write

- **Definition:** Users can write to unintended storage locations.
- **Prevention:** Restrict write access to trusted logic.

49. Upgrade Bricking

- **Definition:** Upgrade logic renders contract unusable.
- **Prevention:** Validate upgrade flow with test suites.

50. Emergency Function Abuse

- **Definition:** Admin emergency features used maliciously.
- **Prevention:** Protect emergency functions with governance.

51. Internal Accounting Drift

- **Definition:** Accounting logic drifts from actual balance.
- **Prevention:** Reconcile state with real balance regularly.

52. Delegatecall to Self

- **Definition:** Calling own logic via delegatecall introduces reentrancy.
- **Prevention:** Avoid unnecessary delegatecalls to self.

53. User-Defined Callback Injection

- **Definition:** Malicious callbacks interfere with core logic.
- **Prevention:** Limit callback use and validate caller identity.

54. Math Underflow in Loops

- **Definition:** Loop counters underflow, causing infinite loops.
- **Prevention:** Use unchecked only when safe; add guards.

55. Faulty Permit Implementation

- **Definition:** EIP-2612 not implemented securely.
- **Prevention:** Follow OpenZeppelin's audited pattern.

56. Insufficient Fee Logic in DEXs

- **Definition:** Protocol fees calculated incorrectly.
- **Prevention:** Test fee path with various edge cases.

57. Premature Selfdestruct Call

- **Definition:** Destruct function accessible before migration.
- **Prevention:** Lock destruct behind finalization phase.

58. Oracle Feed Delay Exploits

- **Definition:** Exploiting delayed oracle updates.
- **Prevention:** Use minimum and maximum price windows.

59. Counter Overflow

- **Definition:** Repeated usage causes counter to wrap.
- **Prevention:** Use safe increment libraries.

60. Reused Hash Collision

- **Definition:** Non-unique hashes allow collisions in mappings.
- **Prevention:** Add salts or domain separators.

61. Early Claiming of Rewards

- **Definition:** Rewards claimed before vesting completes.
- **Prevention:** Check timestamps and vesting logic rigorously.

62. Missing Nonce Checks in MetaTx

- **Definition:** Replay of signed transactions.
- **Prevention:** Always include nonce verification.

63. Unsafe tx.origin Usage

- **Definition:** Auth logic uses tx.origin, allowing phishing.
- **Prevention**: Use msg.sender instead.

64. Looping External Calls

- **Definition:** Loop over external addresses leads to DoS.
- **Prevention:** Avoid unbounded loops with external calls.

65. Unchecked Slippage in Multi-Hop Swaps

- **Definition:** Slippage accumulates across swap steps, leading to large losses.
- **Prevention:** Enforce per-hop minimum return checks.

66. Premature Airdrop Claiming

- **Definition:** Users claim tokens before they’re eligible.
- **Prevention:** Add strict timestamp and Merkle root checks.

67. Inconsistent Decimal Precision

- **Definition:** Arithmetic errors due to mismatched token decimals.
- **Prevention:** Normalize all token amounts using their decimals() value.

68. Ignoring SafeERC20

- **Definition:** Interacting with non-compliant ERC20 without wrappers.
- **Prevention:** Use SafeERC20 for all token interactions.

69. Token Approval Without Spending Limits

- **Definition:** Granting infinite token approval leads to misuse.
- **Prevention:** Use bounded approvals and prompt re-approvals.

70. Improperly Initialized UUPS Proxy

- **Definition:** Proxy is usable before logic contract is set.
- **Prevention:** Lock constructor and require initialize.

71. Minting to Burn Address

- **Definition:** Tokens minted directly to 0x0 are irretrievable.
- **Prevention:** Check target address != zero address.

72. Token Migration Inconsistency

- **Definition:** Migrated balances don’t match source.
- **Prevention:** Validate and hash-check balances before porting.

73. Contract Address Collision

- **Definition:** Deterministic deployments lead to same contract address.
- **Prevention:** Use salts and CREATE2 safely.

74. Ignoring ChainID in Signatures

- **Definition:** Signature valid across forks.
- **Prevention:** Use EIP-712 with explicit chain ID.

75. Transfer Hook Abuse

- **Definition:** beforeTransfer/afterTransfer logic triggers unexpected behavior.
- **Prevention:** Scope side-effects and validate balances carefully.

75. Incorrect Reward Scaling

- **Definition:** Multipliers or dividers misapplied in calculation.
- **Prevention:** Use fixed-point math libraries.

76. Unused Governance Voting Results

- **Definition:** Governance vote passed but not enacted.
- **Prevention:** Validate that results trigger execution.

77. Flawed Migration Contracts

- **Definition:** Tokens or users not correctly migrated.
- **Prevention:** Simulate migrations on testnets.

78. Approval Reset Race Condition

- **Definition:** User resets approval while another tx front-runs.
- **Prevention:** Use permit or increaseAllowance.

79. Replay via Permit Signatures

- **Definition:** Permit signatures used multiple times.
- **Prevention:** Track used signature hashes or nonces.

80. Gas Escalation Griefing

- **Definition:** Malicious actor drives up gas price in loop.
- **Prevention:** Cap retries and track gas used.

81. Front-Running NFT Minting

- **Definition:** Bots predict and mint rare NFTs.
- **Prevention:** Use commit-reveal or randomized assignment.

82. Locked Contract via Unused Modifiers

- **Definition:** Required modifier never set.
- **Prevention:** Test full access paths for functionality.

83. Oracle-Free Arbitrage

- **Definition:** Contract ignores external price movements.
- **Prevention:** Use TWAP oracles or frequent updates.

84. NFT Metadata Tampering

- **Definition:** External metadata URLs are modified to misrepresent token.
- **Prevention:** Use on-chain or immutable IPFS links.

85. Token ID Reuse

- **Definition:** Same NFT ID minted multiple times.
- **Prevention:** Enforce unique ID generation logic.

86. Unverified Upgrade Implementations

- **Definition:** New logic contract is unreviewed or malicious.
- **Prevention:** Require audit and multisig confirmation before upgrades.

87. Gas Token Exploitation

- **Definition:** Refunds used to manipulate execution cost.
- **Prevention:** Disable gas refund logic if not needed.

88. Broken Interface Compatibility

- **Definition:** Interfaces not aligned with actual logic.
- **Prevention:** Verify interface compliance using Solidity tools.

89. Vesting Shortcut Exploits

- **Definition:** Edge cases let users withdraw early.
- **Prevention:** Use block-based vesting with precise boundaries.

90. Read-Only Reentrancy

- **Definition**: View functions cause side effects when combined with fallback logic.
- **Prevention:** Do not rely on read-only context for safety.

91. Message Data Manipulation

- **Definition:** Malicious inputs alter behavior via abi-encoded payloads.
- **Prevention:** Decode inputs and validate boundaries explicitly.

92. Ignoring Transfer Return Status

- **Definition:** ERC20 transfer() fails silently but logic proceeds.
- **Prevention:** Check for return status explicitly.

93. Incentive Misalignment in Staking

- **Definition:** Users game reward mechanics for unfair gains.
- **Prevention:** Simulate edge cases and adjust incentive curves.

94. Whitelist Bypass in Launchpads

- **Definition:** Non-whitelisted users mint tokens.
- **Prevention:** Enforce Merkle root proofs.

95. Flash Loan + Deposit Exploits

- **Definition:** Users use flash loans to inflate deposit rewards.
- **Prevention:** Enforce min holding periods.

96. Bribery Through Vote Delegation

- **Definition:** Delegates get bribed without disclosure.
- **Prevention:** Use off-chain voting transparency tools.

97. Flawed Slashing Conditions

- **Definition:** Honest actors are penalized due to bad logic.
- **Prevention:** Simulate slashing on valid/invalid activity patterns.

98. Stale Signature Usage

- **Definition:** Signatures don’t expire, can be reused later.
- **Prevention:** Enforce expiration timestamps.

99. NFT Royalty Bypass

- **Definition:** Secondary market trades bypass royalty mechanism.
- **Prevention:** Enforce royalty at protocol level.

100. Zero Fee Manipulation

- **Definition:** Users exploit zero fee setting to game reward system.
- **Prevention:** Set min fee boundaries and monitor activity.

101. Shadow Fork Confusion

- **Definition:** Conflicting data from forked chain used in logic.
- **Prevention:** Validate chain ID and block hashes.

102. Loss of Precision in Large Number Operations

- **Definition:** Truncation errors in big integer math.
- **Prevention:** Use libraries like PRBMath for fixed-point precision.

103. Pseudo-Randomness via Wallet Address

- **Definition:** Deriving randomness from msg.sender.
- **Prevention:** Use external randomness like VRF.

104. LP Token Burn With No Lock

- **Definition:** LP tokens burned prematurely without vesting.
- **Prevention:** Lock LP tokens and enforce unlock schedules.

105. Stuck Swap Pools

- **Definition:** Pool gets stuck due to imbalanced reserves.
- **Prevention:** Add emergency rebalancing or auto-arbitrage.

106. Proxy Storage Overlap on Upgrade

- **Definition:** Upgraded logic writes to wrong storage slots.
- **Prevention:** Use structured storage with gap padding.

107. Voting Without Snapshot

- **Definition:** Vote count changes during voting period.
- **Prevention:** Snapshot balances at vote start.

108. Bricked Token Contracts via Total Supply

- **Definition:** Tokens can’t transfer due to improper supply limits.
- **Prevention:** Audit mint/burn/supply math.

109. Loss of Funds via Fee-On-Transfer Tokens

- **Definition:** Swaps fail due to fee logic not accounted.
- **Prevention:** Adjust swap logic to deduct fee-on-transfer behavior.

110. Fund Drain via Malicious Migration

- **Definition:** Migration contract steals tokens.
- **Prevention:** Use audited contracts and verify migration calls.

111. Zero Knowledge Circuit Exploit

- **Definition:** zk verification circuit has logical error.
- **Prevention:** Use audited ZK circuits and test with edge inputs.

112. NFT Replay on Bridged Chains

- **Definition:** NFTs reappear on original chain despite bridging.
- **Prevention:** Burn on origin chain before minting on destination.

113. Liquidity Rug via RevokeAll

- **Definition:** Admin revokes all approvals, halting swaps.
- **Prevention:** Decentralize approval and upgrade flows.

114. Bad Actor in Oracle Committee

- **Definition:** Oracle provider feeds fake data.
- **Prevention:** Use majority oracles and slashing for misbehavior.

115. Fake Contract Interface Abuse

- **Definition:** Contract mimics interface but behaves differently.
- **Prevention:** Verify bytecode of trusted contracts.

116. Reverse Reentrancy via Callbacks

- **Definition**: External contract calls back into view function.
- **Prevention:** Use stateless reads and strict access layers.

117. Loopable

- **Definition:** Logic abusable via loops or proxies.
- **Prevention:** Limit claims per address and detect proxies.

118. NFT Mint Exhaustion

- **Definition:** Max token supply hit but mint logic not updated.
- **Prevention:** Enforce cap at logic and frontend levels.

119. Token Initialization Without Admin

- **Definition:** Tokens deployed without owner control.
- **Prevention:** Require admin and setup before enable.

120. Insecure Upgrade Path in Proxy Pattern

- **Definition:** Anyone can upgrade logic.
- **Prevention:** Restrict upgrade with onlyAdmin or governance.

121. Incorrect Refund on Failed Transaction

- **Definition:** Refunds issued even if action fails.
- **Prevention:** Tie refund to success flag.

122. Chain Reorg Inconsistency

- **Definition:** State changes rolled back in reorg.
- **Prevention:** Finalize only after confirmations.

123. Liveness Exploit in Off-Chain DAOs

- **Definition:** Few active voters allow fast proposal passage.
- **Prevention:** Require quorum or delayed execution.

124. Message Bridge Replay

- **Definition:** Same cross-chain message replayed.
- **Prevention:** Use nonce-based replay protection.

125. Meta-Transaction Relay Abuse

- **Definition:** Relayer alters transaction order or gas fee.
- **Prevention:** Sign full transaction data including nonce and gas.

126. Underflow in Reward Distribution

- **Definition:** Division by zero or subtraction underflows.
- **Prevention:** Always validate divisor and balances.

127. Misconfigured Keepers

- **Definition:** Off-chain keepers update logic incorrectly.
- **Prevention:** Add keeper verification and time checks.

128. Hidden Logic in Proxy Initialization

- **Definition:** Malicious logic embedded in initializer.
- **Prevention:** Isolate initializer logic and audit.

129. Treasury Drain via Governance Attack

- **Definition:** Governance proposal drains protocol funds.
- **Prevention:** Add timelocks and quorum on treasury moves.

130. NFT Metadata Swapping

- **Definition:** Metadata changes post-mint.
- **Prevention**: Freeze metadata or use on-chain storage.

131. Flash Claim Exploit

- **Definition:** Borrowed NFTs claimed and returned in same tx.
- **Prevention:** Enforce ownership duration for claims.

132. Incorrect Reward Indexing

- **Definition:** Rewards assigned to wrong index/round.
- **Prevention:** Validate epoch index and mapping logic.

133. External Function Call to Unchecked Contract

- **Definition**: Calling unknown external contract blindly.
- **Prevention:** Whitelist and verify target contracts.