

# L43: Smart Contract Security

## Module F: Advanced Topics

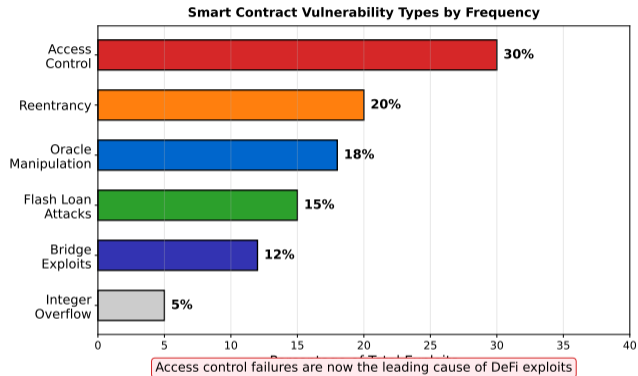
Blockchain & Cryptocurrency Course

December 2025

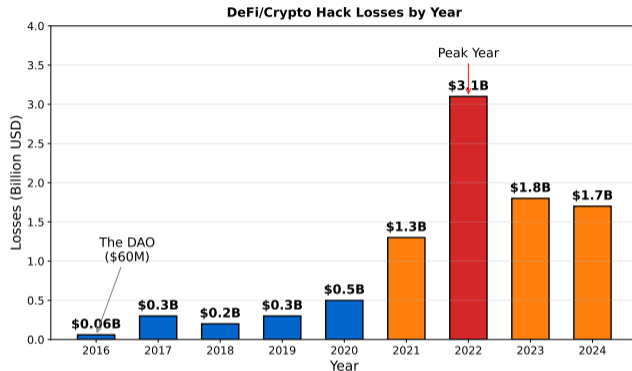
- Understand the stakes of smart contract security
- Identify top vulnerability types (reentrancy, access control, oracle manipulation)
- Analyze real-world exploits (The DAO, Parity, bridge hacks)
- Apply security tools (Slither, Mythril, formal verification)
- Implement defense in depth strategies

- **Code is Law:** Smart contracts are immutable and self-executing
- **High-Value Targets:** DeFi protocols hold billions in assets
- **Irreversibility:** Bugs cannot be patched without upgradeable contracts
- **Losses to Date:** \$8B+ stolen from smart contract exploits (2016-2024)
- **Asymmetry:** One vulnerability can drain entire protocol
- **Composability Risk:** Vulnerabilities cascade across protocols

# Top Vulnerability Types



*Access control failures have become the leading cause of DeFi exploits*



*2022 was peak year (\$3.1B); losses decreasing but still significant*

- **The DAO:** Decentralized organization, \$150M raised
- **Vulnerability:** External call before state update
- **Attack Mechanism:**
  - 1 Attacker deposits 1 ETH
  - 2 Calls withdraw(1), contract sends 1 ETH
  - 3 Attacker's fallback recursively calls withdraw()
  - 4 Balance not yet updated, sends another 1 ETH
  - 5 Loop continues until contract drained
- **Result:** \$60M drained
- **Aftermath:** Ethereum hard fork (ETH vs ETC split)

## Checks-Effects-Interactions Pattern:

- 1 **Check:** Verify conditions (require statements)
- 2 **Effect:** Update state variables FIRST
- 3 **Interaction:** Make external calls LAST

## Reentrancy Guard (Mutex):

- Lock flag prevents recursive calls
- OpenZeppelin ReentrancyGuard is industry standard

**Root Cause:** State updated after external call

**Prevention:** Always update state before external calls

## Common Mistakes

- Missing onlyOwner modifier
- Default function visibility (public)
- Unprotected selfdestruct
- Constructor typo (pre-0.5.0)

## Parity Wallet Hack (2017)

- initWallet() unprotected
- Attacker became owner
- Called selfdestruct
- \$300M frozen forever

## Best Practices

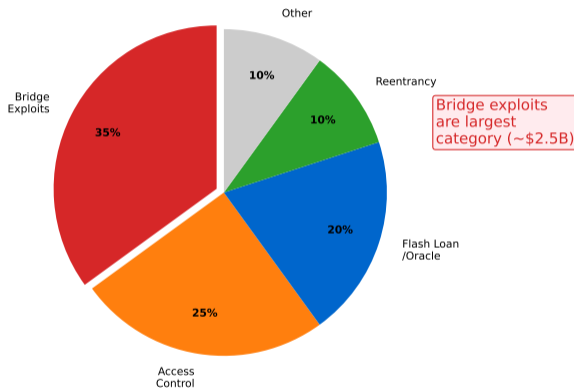
- Use OpenZeppelin Ownable
- Explicit visibility modifiers
- Role-based access control
- Multi-sig for critical functions

## 2024 Reality

- Access control is now top attack vector
- Private key compromises
- Admin privilege escalation

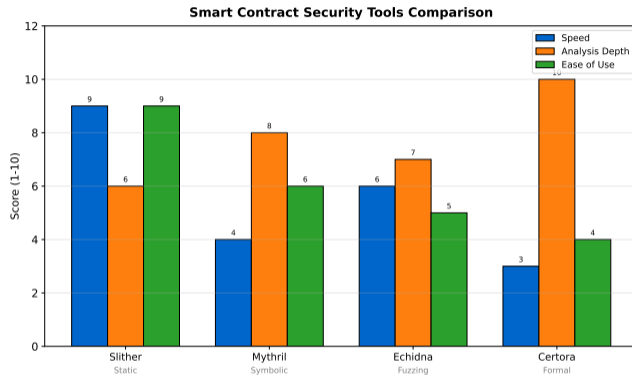
- **Problem:** DeFi relies on external price data (oracles)
- **Vulnerable Pattern:** Using single DEX price as oracle
- **Attack Vector:**
  - 1 Flash loan borrow massive funds
  - 2 Manipulate DEX pool price (10x increase)
  - 3 Protocol reads manipulated price
  - 4 Exploit (borrow, liquidate, mint at wrong price)
  - 5 Restore pool, repay flash loan, keep profit
- **Real Examples:** Harvest Finance (\$34M), Cream (\$130M)
- **Defense:** Chainlink oracles, TWAP, multiple sources

**DeFi Attack Vectors by Total Losses (2020-2024)**



*Bridge exploits account for largest share of total losses*

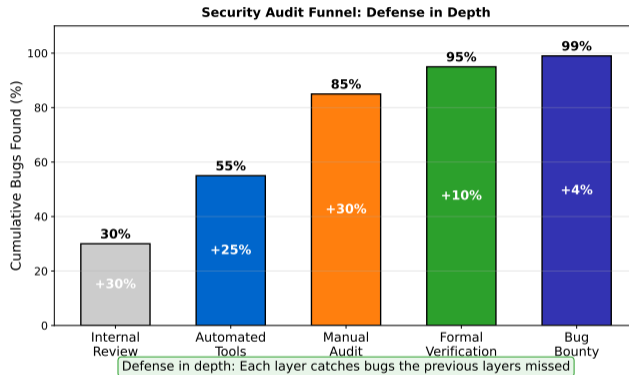
- **Why Bridges?:** Cross-chain transfers require lock-and-mint
- **Attack Surface:** Validators, smart contracts, key management
- **Notable Exploits:**
  - Ronin Bridge (2022): \$625M - validator key compromise
  - Wormhole (2022): \$320M - signature verification bug
  - Nomad (2022): \$190M - initialization vulnerability
- **Key Issues:**
  - Centralized validator sets
  - Complex multi-chain logic
  - High-value targets (\$10B+ locked in bridges)
- **Trend:** Bridge security is top priority for 2024-2025



*Combine fast tools (Slither) with deep analysis (Mythril, Certora)*

Tool	Type	Detects
Slither	Static analyzer	Reentrancy, overflow, access
Mythril	Symbolic execution	Integer bugs, unchecked calls
Echidna	Fuzzer	Invariant violations
Certora	Formal verifier	Specification violations

- **Slither**: Fast, easy CI/CD integration
- **Mythril**: Deep analysis, finds complex bugs
- **Echidna**: Property-based testing
- **Certora**: Mathematical proofs



*Each security layer catches bugs that previous layers missed*

- ① **Internal Review:** Developer self-audit, peer review
- ② **Automated Tools:** Slither, Mythril in CI/CD
- ③ **Manual Audit:** Security firm (2-4 weeks, \$50k-\$500k)
  - Trail of Bits, OpenZeppelin, Quantstamp
- ④ **Formal Verification:** High-value contracts (Certora)
- ⑤ **Bug Bounty:** Community testing (Immunefi)
- ⑥ **Monitoring:** Real-time detection (Forta, OpenZeppelin Defender)
- ⑦ **Insurance:** Coverage for exploits (Nexus Mutual)

- **Purpose:** Incentivize white-hat hackers
- **Platforms:** Immunefi, HackerOne, Code4rena
- **Payouts:** \$1k (low) to \$10M+ (critical)
- **Record:** Wormhole \$10M bounty (2022)
- **Notable Programs:**
  - MakerDAO: Up to \$10M
  - Ethereum Foundation: Up to \$250k
  - Compound: Up to \$500k
- **ROI:** \$1 in bounties prevents \$100+ in losses
- **Best Practice:** Continuous bounty, not just pre-launch

- **Problem:** Immutable contracts cannot be patched
- **Solution:** Proxy pattern (separate storage and logic)
- **Transparent Proxy:**
  - Proxy holds storage, delegates to implementation
  - Admin can upgrade implementation address
- **UUPS (Universal Upgradeable Proxy):**
  - Upgrade logic in implementation (smaller proxy)
- **Risk:** Admin key compromise → malicious upgrade
- **Mitigation:** Multi-sig, timelock delays, immutable after maturity

- 1 Use latest Solidity (0.8.0+ for overflow protection)
- 2 Follow Checks-Effects-Interactions pattern
- 3 Apply reentrancy guards (OpenZeppelin)
- 4 Explicit visibility for all functions
- 5 Decentralized oracles (Chainlink) or TWAP
- 6 Pull over push for payments
- 7 Avoid loops over unbounded arrays
- 8 Run Slither + Mythril in CI/CD
- 9 Professional audit before mainnet
- 10 Bug bounty program
- 11 Real-time monitoring (Forta)

## Key Takeaways:

- Smart contract security is critical: \$8B+ cumulative losses
- Top vulnerabilities: Access control, reentrancy, oracle manipulation, bridges
- The DAO hack (\$60M) led to Ethereum hard fork
- Bridge exploits are now largest loss category
- Defense in depth: Automated tools + audits + formal verification + bounties
- Tools: Slither (fast), Mythril (deep), Certora (formal proofs)
- Upgradeable contracts: Allow patches but introduce admin key risk
- 2024 trend: Real-time monitoring and “Security as a Service”

- ❶ Why is the Checks-Effects-Interactions pattern important?
- ❷ How do oracle manipulation attacks exploit DeFi protocols?
- ❸ What makes bridge security particularly challenging?
- ❹ Should audited protocols still have bug bounties?
- ❺ What are the trade-offs of upgradeable contracts?