Smart contract security, in plain terms.

Smart contracts are unforgiving: the code is public, you usually cannot patch it after deployment, and it controls funds directly. This section breaks the code-level vulnerabilities (reentrancy, integer overflow, access control, delegatecall) and the economic attacks (flash loans, oracle manipulation, front-running, rug pulls) into plain-language explainers, each ending with how an audit catches the issue before it ships.

HOW IT WORKS

01 Key vulnerabilities explained

Plain-language definitions of the bugs and attacks smart contract audits look for. Each page covers what it is, how the attack works, a code example, and how to defend.

Code-level vulnerabilities

- What is a reentrancy attack?
- What is integer overflow and underflow?
- What is an access control vulnerability?
- What is a delegatecall vulnerability?
- What is an unchecked external call?
- What is tx.origin authentication?
- What is a proxy storage collision?

Economic and protocol attacks

- What is a flash loan attack?
- What is oracle manipulation?
- What is front-running and MEV?
- What is a rug pull?
- What is a signature replay attack?
- What is a smart contract denial of service?

02 How to read this section

The pages split into two families.

- Code-level vulnerabilities: bugs in the contract logic itself, reentrancy, integer overflow, access control, delegatecall, unchecked calls, tx.origin, and proxy storage collisions.

SOURCES

[1] OWASP Smart Contract Top 10

[2] Ethereum.org: Smart contract security

[3] Solidity docs: Security considerations

- Economic and protocol attacks: abuses of how the protocol and the wider DeFi ecosystem behave, flash loans, oracle manipulation, front-running and MEV, rug pulls, signature replay, and denial of service.

Each explainer ends with how a smart contract audit catches the issue before the code is deployed and irreversible.

Get your smart contracts audited before they go on-chain.
securelayer7.net/learn/smart-contract-security

Open online