What is an access control flaw?

An access control vulnerability in a smart contract is a privileged function that fails to verify the caller, so anyone can call something only an owner or admin should, minting tokens, withdrawing funds, changing critical parameters, or taking ownership. Causes include a missing modifier, a wrong check, an unprotected initializer, or a public function that should be internal. Because every function is callable by anyone on-chain, an unguarded sensitive function is directly exploitable. The fix is consistent, correct authorization on every privileged path.

HOW IT WORKS

01 How it works and example

A mint function missing its restriction:

```
function mint(address to, uint amount) public {  // no
access control
  _mint(to, amount);
}
```

Anyone calls `mint` and creates unlimited tokens for themselves. Other common cases:

- An unprotected `initialize()` on an upgradeable contract, an attacker calls it to become owner.
- A `selfdestruct` or `withdraw` left public.
- Ownership transfer with a flawed check.

The attacker simply invokes the function directly. Documented for defensive context.

> EVERYTHING IS CALLABLE
>
> *There is no "internal-only by being obscure" on-chain, every public/external function is reachable by anyone. A sensitive function without an explicit, correct authorization check is effectively public.*

HOW TO DEFEND

- Add authorization to every privileged function (an `onlyOwner`/role modifier), and verify the check is correct, not just present.
- Protect initializers on upgradeable contracts so they can be called only once, by the deployer.
- Use a vetted access-control library (role-based access control) rather than ad-hoc checks.
- Make functions `internal`/`private` when they should not be externally callable.
- Audit the full permission model and test that restricted functions reject unauthorized callers.

SOURCES

[1] OWASP Smart Contract Top 10

[2] SWC Registry: Smart Contract Weakness Classification

[3] MITRE CWE-284: Improper Access Control