# Smart Contract Security

## Best Practice Guidelines

Stefan Beyer, Philip Stanislaus and Monty

**OAK Security**

# Who are we?





## Oak Security

- Blockchain Security Specialists
- Focus on third-generation blockchains
- Partnership with Terraform Labs
- Auditors of Columbus-5, Anchor, Mirror and Nebula
- Provider for Terraform Capital funded audits

# What’s wrong with current DeFi security?

# A DeFi Project Timeline Example

Tuesday Evening

Smart Contracts Deployed

Wednesday Afternoon

$580 million in Smart Contracts

Wednesday Evening

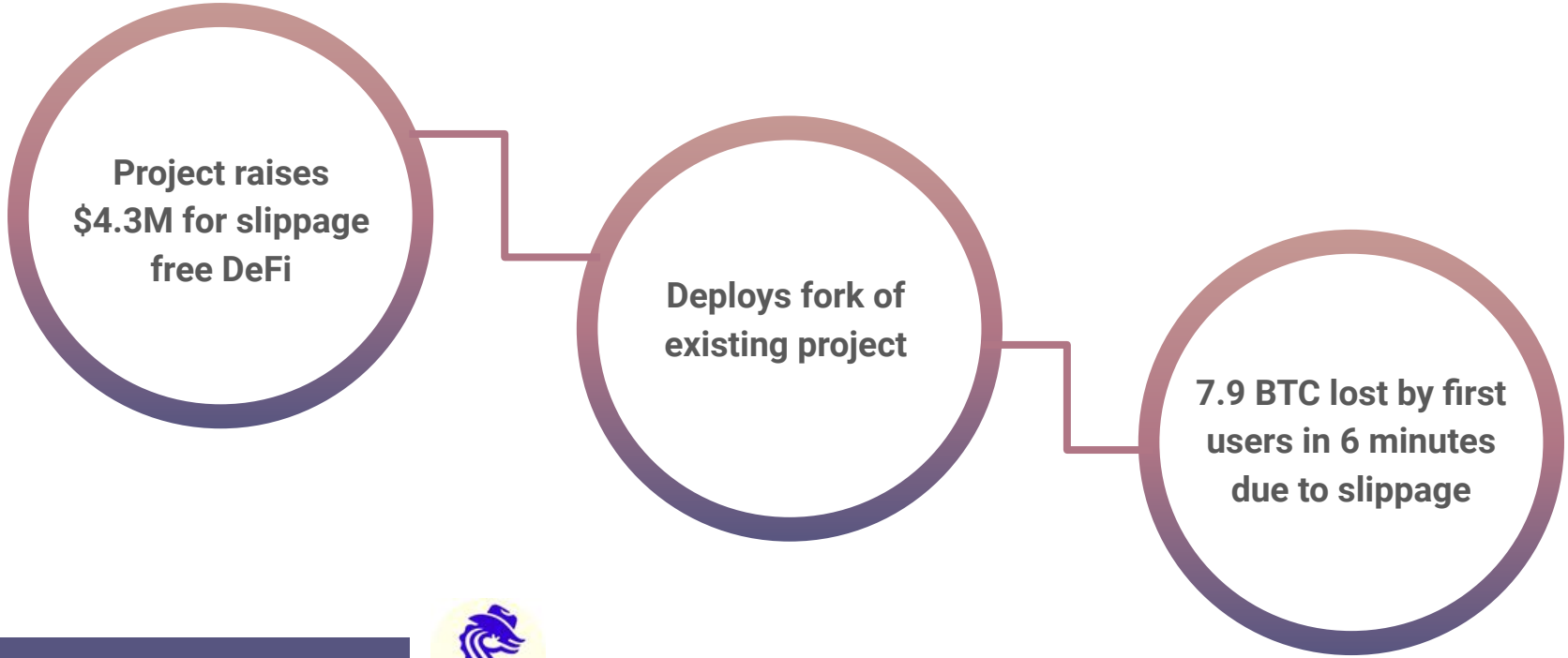
Bug Discovered

Thursday

Price Collapse

This really happened!

# Another Example




This also happened!

# How to build secure smart contract protocols?

# Realize you are building financial software

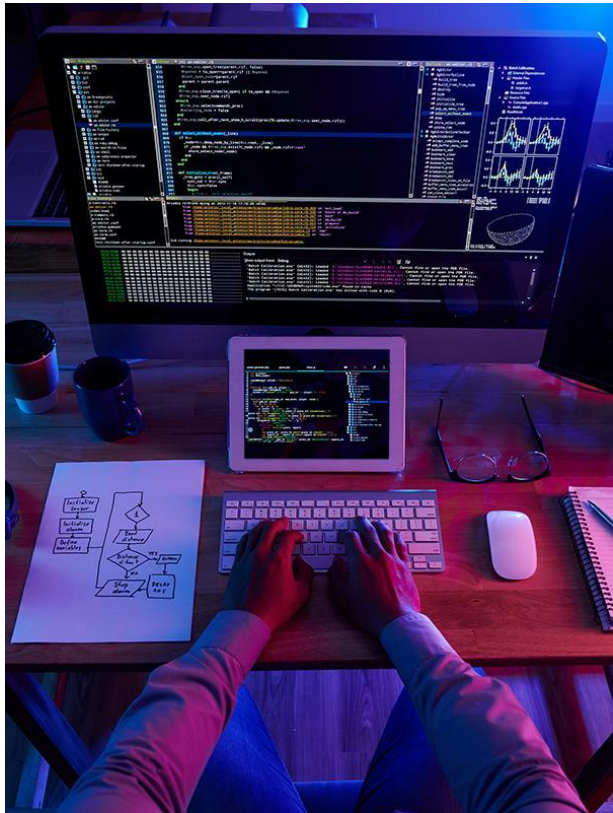

1. Get domain expertise
2. Specify and document
3. Slow Down
4. Test
5. Keep it simple
6. Incremental Innovation

# Best Practice Guidelines

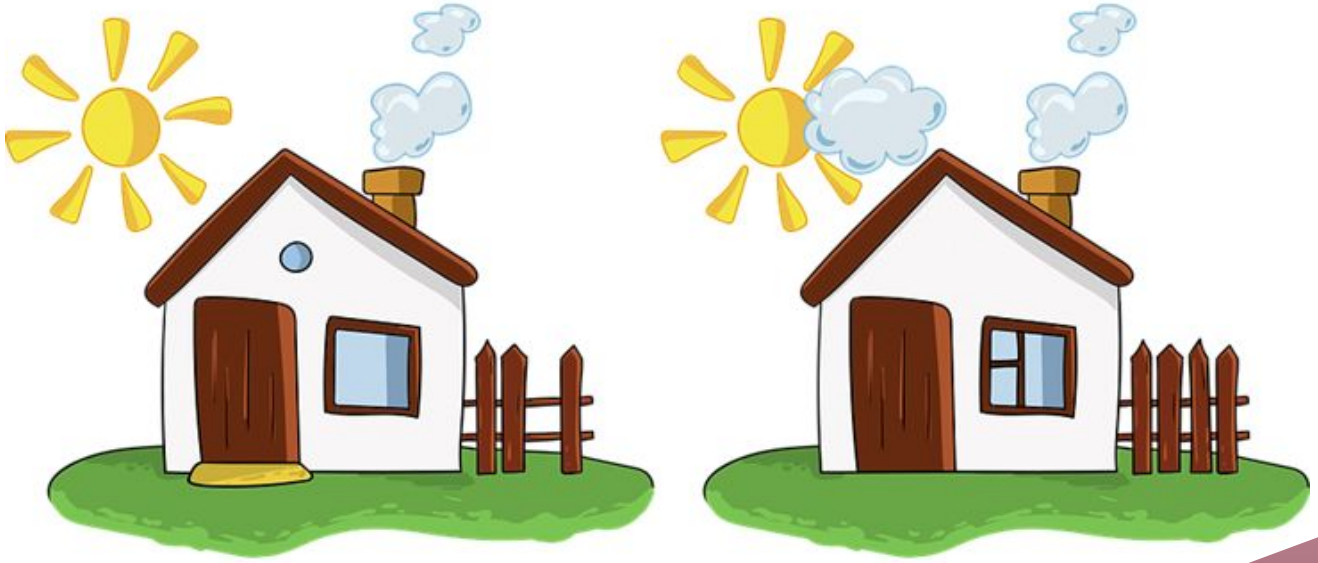

OAK

# Make sure your model is sound

- **A specification essential**
- **Think about edge cases**
  - Large numbers?
  - Small numbers?
  - Big differences between numbers?
  - Weird use cases: Eg. 0-block staking
- **Can the model be gamed?**
  - Misaligned incentives
  - Frontrunning
  - DoS
- **What external factors does your model depend on?**



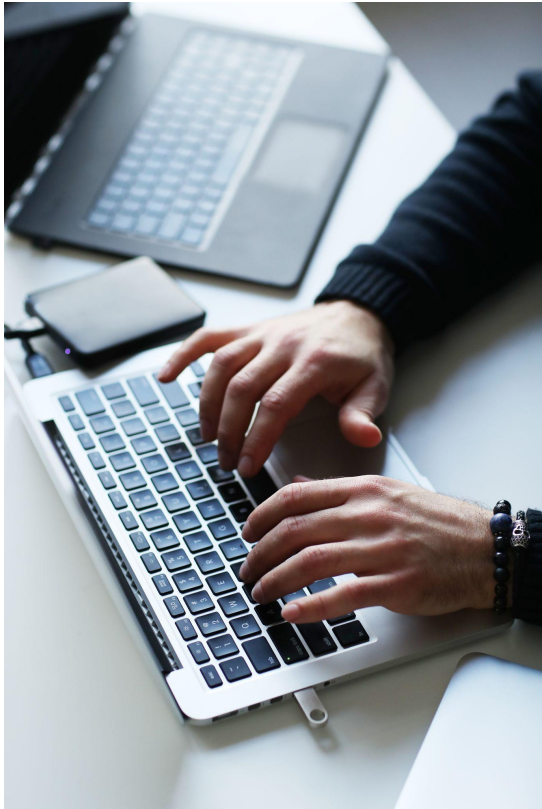
# Implement the Specification!!!



Spot the Difference

# Composability

- **What does your protocol depend on?**
  - Other protocols?
  - Oracles?
- **Never trust external calls!!!!!**
- **What could you be used in conjunction with your protocol?**
- **What would happen to other protocols if they build on your protocol?**
  - Avoid unexpected behaviour
  - Follow the standards
  - Don’t follow the standards if you protocol acts differently



# The limitations imposed by the Blockchain

- **Everything is public → Mempool == Dark forest**
- **Everything has a cost**
  - Avoid unbounded loops
  - Try to stick O(1) operations → data-structures
- **Everything is deterministic**
- **Everything runs in a transaction**

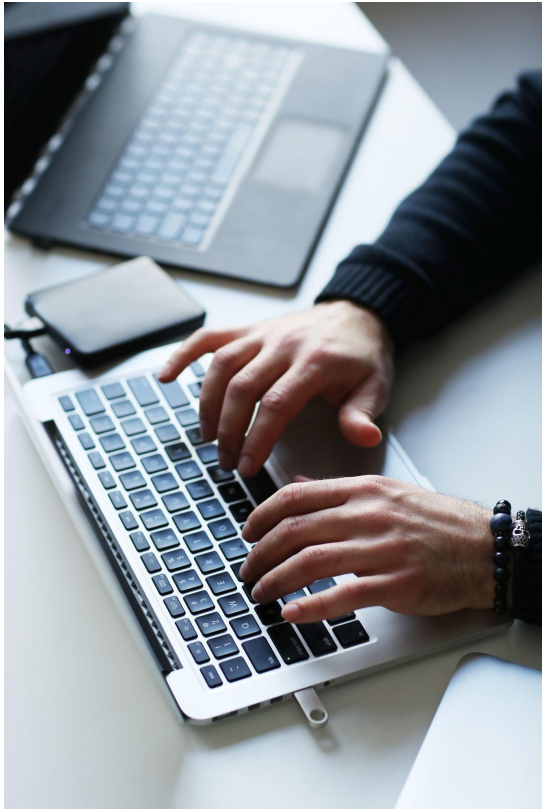
# Trade-offs

- **Immutability vs upgradability**
- **Trustless vs centralized**
  - Open to all assets or whitelisted
  - Governance
  - Privileged accounts
- **Failsafes?**
  - Freeze
  - Timelock

# Testing

- Unit testing
- Fuzz testing
- Integration testing
- UI testing

# Code quality is the first step to security



- Apply formatting (e.g.: rustfmt: cargo fmt)
- Run a linter for example clippy (Rust) or solhint (Solidity)
- Ensure all tests of your projects pass
- Ensure you have a sufficient test coverage, e.g. through tarpaulin or solidity-coverage
- Document the codebase
- Have clear descriptive code comments

# Common pitfalls


- **Authorization** and access levels in contract functions. Seems obvious but even top projects fall into this. Example: leaving as public functionality that's meant to be handled only by the admin.

- **DoS attacks.** Relevant in external calls that may revert or when unbounded loops are depended on user input which can cause out-of-gas problems and locked funds.

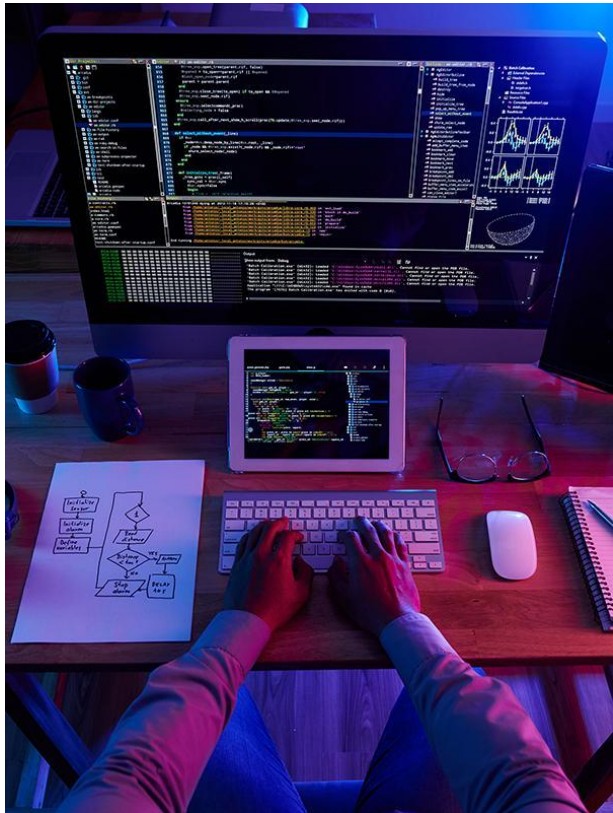
- **Incorrect accounting.** Often related to fees, gas or taxes (e.g. on Terra). Can lead to skimming and run-out-of-funds issues.

- **Insecure Randomness.** Missing Entropy in blockchains means randomness hard to achieve. Common pseudo random sources are public before transactions confirmed or can be manipulated.

- **Composability.** Lack of understanding of the contracts your project is integrating with can create vulnerabilities.

# Operational Security

- Do you trust your team?
- Who controls your keys? → Use multisig
- How do you communicate?
- How do you give and revoke access?
- Have contingency plans for all occurrences
- Stay up to date with platform and infrastructure updates



# Question?


@SecurityOak


https://oaksecurity.io


https://github.com/oak-security


info@oaksecurity.io