15 lines of code that could have prevented TheDAO Hack

OPENZEPPELIN SECURITY | OCTOBER 5, 2016

Security Audits

by Manuel Araoz

As you probably know, TheDAO hack was a tragic event for the Ethereum community. $50M USD were lost from the original contract, ETH price crashed, a hard fork was conducted as a “fix”, which spawned Ethereum Classic into existence, and many people lost confidence in blockchains as immutable platforms where is law.

On the bright side, this sparked really great conversations about smart contract security. New really cool projects emerged trying to improve the tools available to smart contract developers (one of which is OpenZeppelin).

The biggest problem with TheDAO was that it was vulnerable to reentrancy. This allowed the attacker to execute malicious in between the contract’s execution, and steal funds. A good pattern that prevents that problem is doing pull payments instead of push payments

This has been discussed by the Ethereum developer community for a while, so I decided to write a helper contract anyone can use to adopt this pattern easily. Just make your contracts inherit from this contract and replace *send* calls with *asyncSend:*

```
/*
* PullPayment
```

```
contract

PullPayment {

mapping
(address => uint) public payments;

  // store sent amount as credit to be pulled, called by payer

function
 asyncSend(address dest, uint amount) internal {
    payments[dest] += amount;
  }

  // withdraw accumulated balance, called by payee

function
 withdrawPayments() external {
    uint payment = payments[msg.sender];
    payments[msg.sender] = 0;
    if (!msg.sender.send(payment)) {
      payments[msg.sender] = payment;
    }
  }
}
```

Every time you call *asyncSend*, balance is added to the caller's credit. Note that the function is internal and can only be called by this contract or contracts inheriting from it. Once someone has balance, they can call the *withdrawPayments* function to get the actual payment sent.

This decoupling of payment from the rest of the contract logic isolates the *send* call in such a way that each payee can't mess with the rest of the funds or . If you want to see an example usage of this class, check out this sample bid contract

OpenZeppelin

Related Posts

Beefy Zap Audit

BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

Security Audits

OpenBrush Contracts Library Security Review

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Security Audits

Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits

OpenZeppelin

OpenZeppelin

Threat Monitoring
Incident Response
Operation and Automation

Zero Knowledge Proof Practice

Blog

Company

About us
Jobs
Blog

Contracts Library

Docs