Designing (More) Secure Smart Contracts

Nemil Dalal

Stanford CS359B

May 16, 2018

DAO Hack

```
function splitDAO(
  uint _proposalID,
  address _newCurator
) noEther onlyTokenholders returns (bool _success) {

  ...
  // XXXXX Move ether and assign new Tokens.  Notice how this is done first!
  uint fundsToBeMoved =
      (balances[msg.sender] * p.splitData[0].splitBalance) /
      p.splitData[0].totalSupply;
   // XXXXX This is the line the attacker wants to run more than once
  if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false) e
      throw;

  ...
  // Burn DAO Tokens
  Transfer(msg.sender, 0, balances[msg.sender]);
  withdrawRewardFor(msg.sender); // be nice, and get his rewards
  // XXXXX Notice the preceding line is critically before the next few
  totalSupply -= balances[msg.sender]; // XXXXX AND THIS IS DONE LAST
  balances[msg.sender] = 0; // XXXXX AND THIS IS DONE LAST TOO
  paidOut[msg.sender] = 0;
  return true;
}
```

$150 MM+ from 11k addresses, 15% of all Ether

Source: http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/

King of the Ether Throne

```
1   contract KotET {
2     address public king;
3     uint public claimPrice = 100;
4     address owner;
5
6     function KotET() {
7       owner = msg.sender;  king = msg.sender;
8     }
9
10    function sweepCommission(uint amount)  {
11      owner.send(amount);
12    }
13    function() {
14      if (msg.value < claimPrice) throw;
15
16      uint compensation = calculateCompensation();
17      king.send(compensation);
18      king = msg.sender;
19      claimPrice = calculateNewPrice();
20    }
21    /* other functions below */
22  }
```

Source: https://applicature.com/blog/history-of-ethereum-security-vulnerabilities-hacks-and-their-fixes

parity

technologies ltd

Step 1

```
  function initWallet(address[] _owners, uint _required, uint _daylimit)
only_uninitialized {
    initDaylimit(_daylimit);
    initMultiowned(_owners, _required);
  }
```

Step 2

```
  function kill(address _to) onlymanyowners(sha3(msg.data)) external {
    suicide(_to);
  }
```

$150 MM+ locked

Source: https://elementus.io/blog/which-icos-are-affected-by-the-parity-wallet-bug/

Quick Background

Stanford Electrical Eng/Econ and MBA

Worked on financial IT with top US Banks
as management consultant

Consultant at Gates Foundation's
Financial Services for the Poor

Co-author of Consensys *Smart*
Contracts Best Practices Guide (2016)

YCombinator Founder

Founder/CEO, CryptoFin
(financial products and auditing)

Agenda

1. How is smart contract development different
2. General principles of smart contract development
3. Concrete Tips

Traditional Web Dev	**Smart Contract Dev**
Continuous deployment	Single deployment (and upgrading is hard)
Mature norms in Node/Ruby/Python	Still developing norms
Blind reliance on libraries	Risk from all dependencies
Code often private	Code often publicly available
"Move fast and break things"	***"Move slow and get things right"***

Cost to failure in Software

Lower cost

(lot of Silicon Valley software)

Higher cost

(lots of money or where humans involved)

~~Goal~~

Bug free smart contracts

(Sorry, your contracts will have bugs)

You have finite time to get contract secure,
but an attacker has infinite time to find bugs

More Realistic Goal

Resilient Smart Contracts

Common Issues

Reentrancy

Unsafe Math

Too broad function visibility

Low level calls without safety checks

Blind use of external dependencies

Bad Randomness

Unbounded Iteration

https://dasp.co/

General Mindset

Keep contracts simple

Prepare for Failure

Rollout in stages

Leverage community tools

Stay up to date

Keep Contracts Simple

Use blockchain where relevant

Minimize external functions

Isolate critical functionality into separate contract

Reduce LOC

Prepare for Failure

Speed bumps

Circuit breakers

Assert checks

Emergency roles

Rollout in stages

Expiry locks

Bug bounty on testnet

Third-party Audit

Delineate base vs top layer

Leverage Community Tools

OpenZeppelin

Trail of Bits

Mythril

Stay up to date

Hacking Distributed Blog

dasp.co

Key Smart Contract Security Researchers

Reddit

Chat rooms

A final dose of pragmatism for startups

Security is not the only
optimization criteria in software
e.g, dev productivity, speed to market, usability, gas costs

… but, you can get a lot of smart contract security without tremendous effort.

CryptoFin

nemild@cryptofin.io

Telegram	t.me/cryptofinlabs
Personal Blog	nemil.com/musings

Appendix

All contracts are not created the same

ERC721 Collectible

$\delta Y / \delta X$

Lower Complexity — *Higher Complexity*

short contract (200-1k LOC)

limited user actions

Tens of thousands of $ max

Many millions $ in anticipated volume

Financial products

1000s of LOC

Many external actions

Things we look for in Tests

Unit tests

Integration tests

Good test coverage

Extreme test scenarios

Tests for all dependencies