Smart Contract Security

Christoph Jentzsch - 21.09.2016

Devcon2

The Heist

On 17th of June an attacker tried to rob ~3.5M ETH using the reentry exploit published by Christian Reitwiessner:

```
// THIS CONTRACT CONTAINS A BUG - DO NOT USE
contract Fund {
    /// Mapping of ether shares of the contract.
    mapping(address => uint) shares;
    /// Withdraw your share.
    function withdraw() {
        if (msg.sender.call.value(shares[msg.sender])())
            shares[msg.sender]  = 0;
    }
}
```

```
contract Recipient {
    uint counter;
    function() {
        if (counter < 10) {
            Fund(msg.sender).withdraw();
            counter+=1;
        }
    }
}
```

The Heist

```
function splitDAO(...
    ...
    withdrawRewardFor(msg.sender); // be nice, and get his rewards
    totalSupply -= balances[msg.sender];
    balances[msg.sender] = 0;
    paidOut[msg.sender] = 0;
    return true;
}
```

Smart Contract Security

Cap contracts

- It’s early days - we lack experience:
  - Solidity version 0.4.2
  - Mist version 0.8.2
  - Geth version 1.4.12
  - Frontier has been launched ~ 1 year ago
  - Number of operating Dapps still very low

- Vitalik suggested 10M$ as cap in foundation blog

Formal proof verification

Mathematically proof that a contract has a certain feature or invariant

http://dr-y.no-ip.net/

Invariant Checks

Can be placed in every function:

E.g.: `totalSupply <= this.balance + totalRewardToken`

```
function checkInvariants {

        if (totalSupply > this.balance + totalRewardToken) throw;

    }

function doSomething() {

     …

     checkInvariants();

}
```

Centralization

Going stepwise from centralization to decentralization

- Ethereum: Olympic - Frontier (canaries) - Homestead (difficulty increase) - Metropolis …
- DAO: Curators (except of “splitDAO”)
- DigixDAO, MakerDAO

Who could control it:

- token holders (The DAO)
- central trusted authority (DigixDAO)
- “Community multisig” ?
- Stake Vote (X% of all Ether)

Establish security patterns

- 1024 call stack depth -> always check return values of each call
- Block gas limit -> No arbitrary length loops
- Reentry exploit -> update state **before** executing CALLs
- Ether sent to contract without contract invocation -> be careful with Invariants
- Specify right amount of gas (SEND vs CALL)
- Block timestamp can be manipulated -> block.number are safer
- Tx.orgin vs msg.sender (pishing attacks)
- …

Literature: https://github.com/ConsenSys/smart-contract-best-practices

http://solidity.readthedocs.io/en/latest/security-considerations.html

Updateable contracts

The DAO: “updateContract(address _newContract)” through vote

- Did need 2 weeks debating time
- DAO 1.1 was work in progress

Who can update it:

- token holders (The DAO)
- central trusted authority (DigixDAO)
- “Community multisig” ?
- Stake Vote (X% of all Ether)

Time Delays

DAO:

- 7 Days for splitDAO proposals
- 14 Days for regular proposals
- 27 days creation period
- …

Gives time for a central authority (if implemented in the contract) to act

Minimal complexity

Statistics: ~15-50 bugs per 1000 lines of code

Not everything needs decentralization and needs to be in the smart contract

- Only include in a smart contract the very core of a Dapp

- Reuse trusted proven code
  - Standard Token Contract
  - Foundation multisig
  - (Hopefully one day a DAO standard framework)

Better tools

- Formal proof verification (work in progress)
- Compiler warnings (work in progress)
- Improved IDEs (work in progress)
- Trusted Libraries (work in progress)
- Best practices literature (work in progress)
- Decentralized master keys / Decentralized escape hatches / trusted community multisig to be used in smart contracts as centralized authorities

Conclusion

It’s early days, be careful!