Smart Contract Security

...

- Satheesh Kathamuthu

Agenda

- Smart contracts
- History
- Types of attack vectors
- Writing secure smart contracts
- Conclusion

Smart Contracts

Smart contracts are stateful executable objects hosted on blockchains like Ethereum; carry billions of dollars worth of coins and cannot be updated once deployed.

- Immutable
- “Code is law”

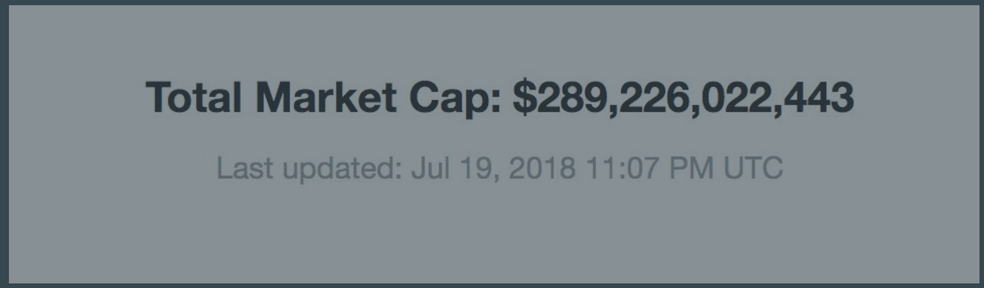
Some statistics

E&Y Study results [6]

- US$ 3.7 Billion raised from ICO
- US$ 400 Million lost or stolen (~10 % of total)
- > 70% of the ICOs included in the study were based on Ethereum

6. E&Y Big Risk in ICO

Live Statistics:



- 34,000 Ethereum contracts are vulnerable [1]
- Ξ 4905 (at Risk due to vulnerable Ethereum contracts

Source: 1. Finding The Greedy, Prodigal, and Suicidal Contracts at Scale arXiv:1802.06038v2 [cs.CR]

2. https://coinmarketcap.com/

What history teaches us ?

- The DAO - Ξ 3.5 M == US$ 50M
- Parity MultiSig - Ξ 150K = US$ 31M lost in Parity Hack; WhiteHat saved the rest (US$ 150 M saved)
- Governmental (1100 ETH stuck because payout exceeds gas limit)
- 5800 ETH swiped (by whitehats) from an ETH-backed ERC20 token
- The King of the Ether game
- Rubixi : Fees stolen because the constructor function had an incorrect name, allowing anyone to become the owner

Types of attacks

- Re-entrancy
- Call to the unknown
- Gasless send
- Keeping secrets
- Front running (Transaction order dependency)
- Stack size limit

Re-entrancy : the DAO, Maker's ETH-backed token

Sends failing due to 2300 gas limit: King of the Ether

Arrays/loops and gas limits: GovernMental

Variable/function naming mixups: FirePonzi, Rubixi

Staying secure

Prepare for failure - at any moment, in any contract or method

Rollout carefully - bug bounties before ICO

Keep contracts simple; more complexity = more attack vectors

Don’t write fancy code

Use audited and tested code

Write as many unit tests as possible

Keep updating with software and community

Beware of blockchain properties: public vs. private, .send() vs. .call()

Conclusion

- blockchain programming vs traditional programming
- Smart contract development toolchain
- Evolution of smart contract ecosystem
- Strength is weakness
- OpenZeppelin

Q&A

...

Thank you !

...

Apendix

...

Re-entrancy

```
2   // INSECURE
3   mapping (address => uint) private userBalances;
4
5   function withdrawBalance() public {
6       uint amountToWithdraw = userBalances[msg.sender];
7       require(msg.sender.call.value(amountToWithdraw)());
8       // At this point, the caller's code is executed, and can call withdrawBalance again
9       userBalances[msg.sender] = 0;
10  }
11
12  mapping (address => uint) private userBalances;
13
14  function withdrawBalance() public {
15      uint amountToWithdraw = userBalances[msg.sender];
16      userBalances[msg.sender] = 0;
17      require(msg.sender.call.value(amountToWithdraw)());
18      // The user's balance is already 0, so future invocations won't withdraw anything
19  }
```

Favor pull over push for external calls

```
// bad
contract auction {
    address highestBidder;
    uint highestBid;

    function bid() payable {
        require(msg.value >= highestBid);

        if (highestBidder != 0) {
            highestBidder.transfer(highestBid);
            // if this call consistently fails, no one else can bid
        }

        highestBidder = msg.sender;
        highestBid = msg.value;
    }
}
```

```
// good
contract auction {
    address highestBidder;
    uint highestBid;
    mapping(address => uint) refunds;

    function bid() payable external {
        require(msg.value >= highestBid);

        if (highestBidder != 0) {
            refunds[highestBidder] += highestBid;
            // record the refund that this user can claim
        }

        highestBidder = msg.sender;
        highestBid = msg.value;
    }

    function withdrawRefund() external {
        uint refund = refunds[msg.sender];
        refunds[msg.sender] = 0;
        msg.sender.transfer(refund);
    }
}
```

Reference

1. Finding The Greedy, Prodigal, and Suicidal Contracts at Scale arXiv:1802.06038v2 [cs.CR]
2. https://consensys.github.io/smart-contract-best-practices/
3. https://blog.ethereum.org/2016/06/19/thinking-smart-contract-security/
4. Atzei N., Bartoletti M., Cimoli T. (2017) A Survey of Attacks on Ethereum Smart Contracts (SoK). In: Maffei M., Ryan M. (eds) Principles of Security and Trust. POST 2017. Lecture Notes in Computer Science, vol 10204. Springer, Berlin, Heidelberg
5. https://www.reddit.com/r/ethereum/comments/4omdlf/to_kickstart_the_building_safer_smart_contracts/
6. Big Risk in ICO Market https://www.ey.com/Publication/vwLUAssets/Big_Risks_in_ICO_market/%24FILE/1801-2546596%20ICO%20market%20infographic%20v11_SCORE%20Approved_FINAL_JAN%2017%202018.pdf