Security with Smart Contracts

By
Shrutirupa Banerjiee(@freak_crypt)

Who am I??

- Security Consultant
- Blockchain Enthusiast
- Mathematics and Core cryptography researcher

Blockchain

- Shared ledger
- Storing information in a decentralised peer-to-peer network
- Immutable

Ethereum

- Public blockchain

Smart Contract

- A program
- Self Enforcing Agreement

Solidity Programming

- HIgh Programming Language to write Ethereum Based Smart Contracts
- Compiled to EVM(Ethereum Virtual Machine) Bytecode

```
pragma solidity ^0.4.22;

contract helloWorld {
 function renderHelloWorld () public pure returns (string) {
   return 'helloWorld';
 }
}
```

What do we need to know??

- Ether
- Gas
- EVM
- Opcodes and a lot more

Testing Smart Contracts

Why???

- once deployed on the blockchain they become immutable
- an application may be critical
- very difficult to update an application once deployed

How???

- Deploy the contract to live (the real) Ethereum main network and execute it.
- Deploy the contract to the test-net Ethereum network and execute it.
- Deploy the contract to an Ethereum network locally and execute it.

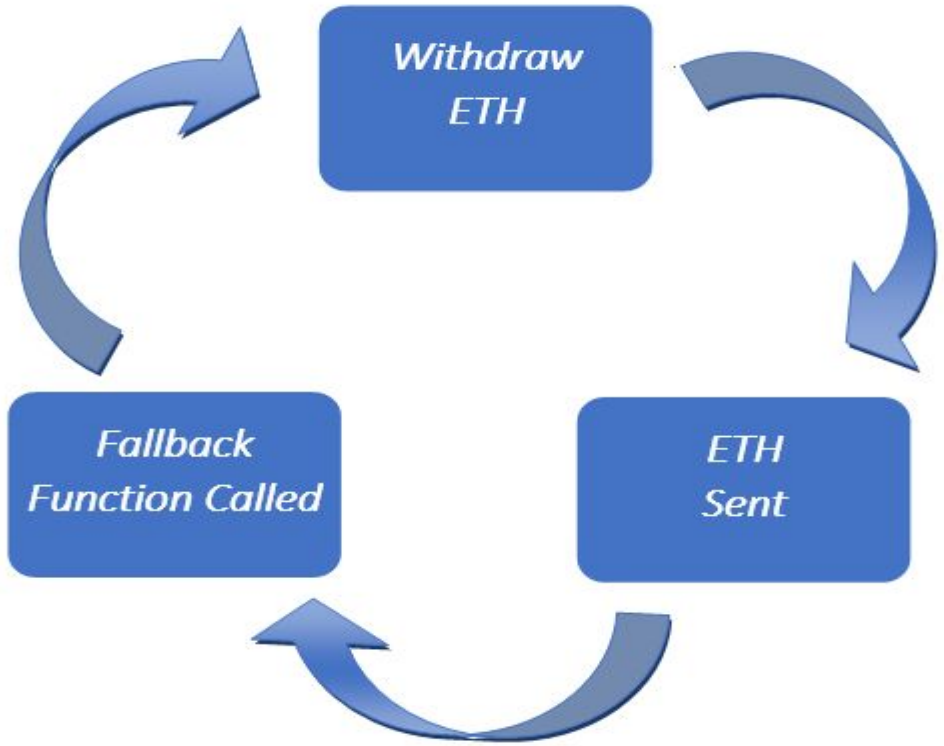
What should we look for???

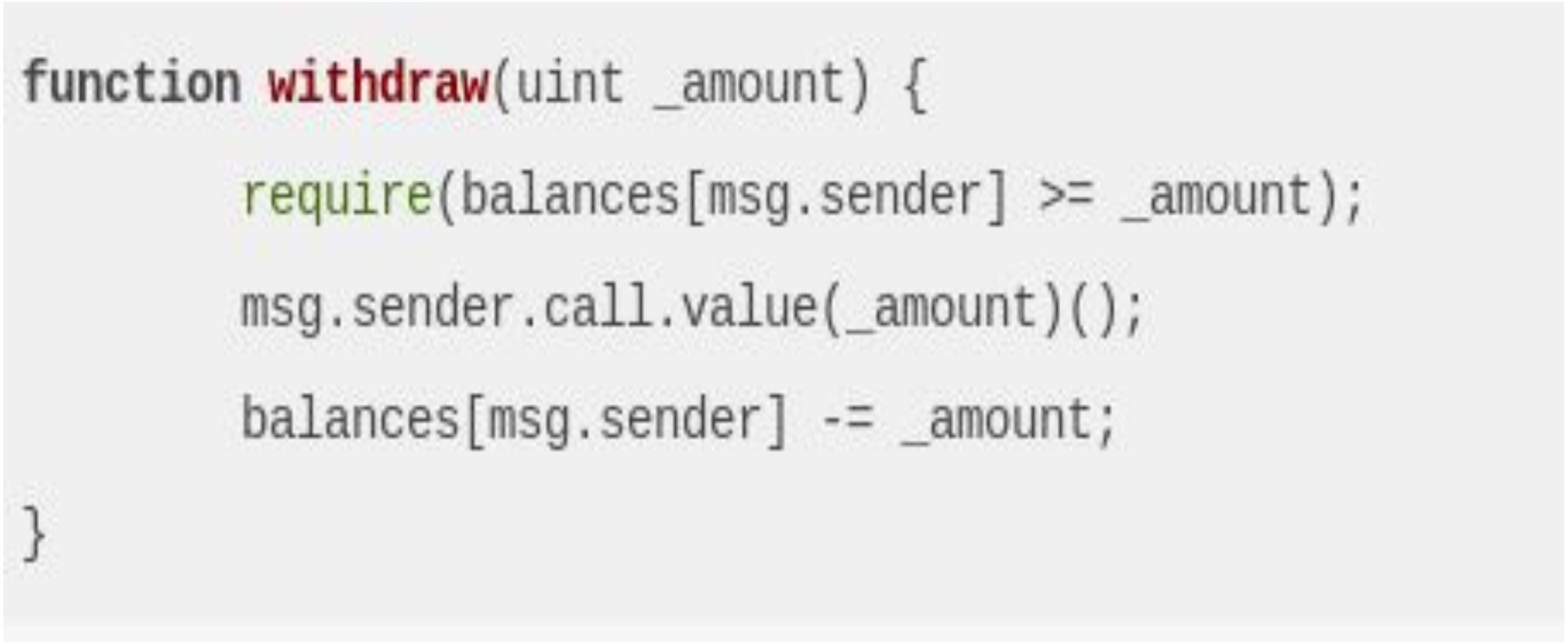
- Already known bugs

- Some unusual behaviour

Let us explore some already known bugs

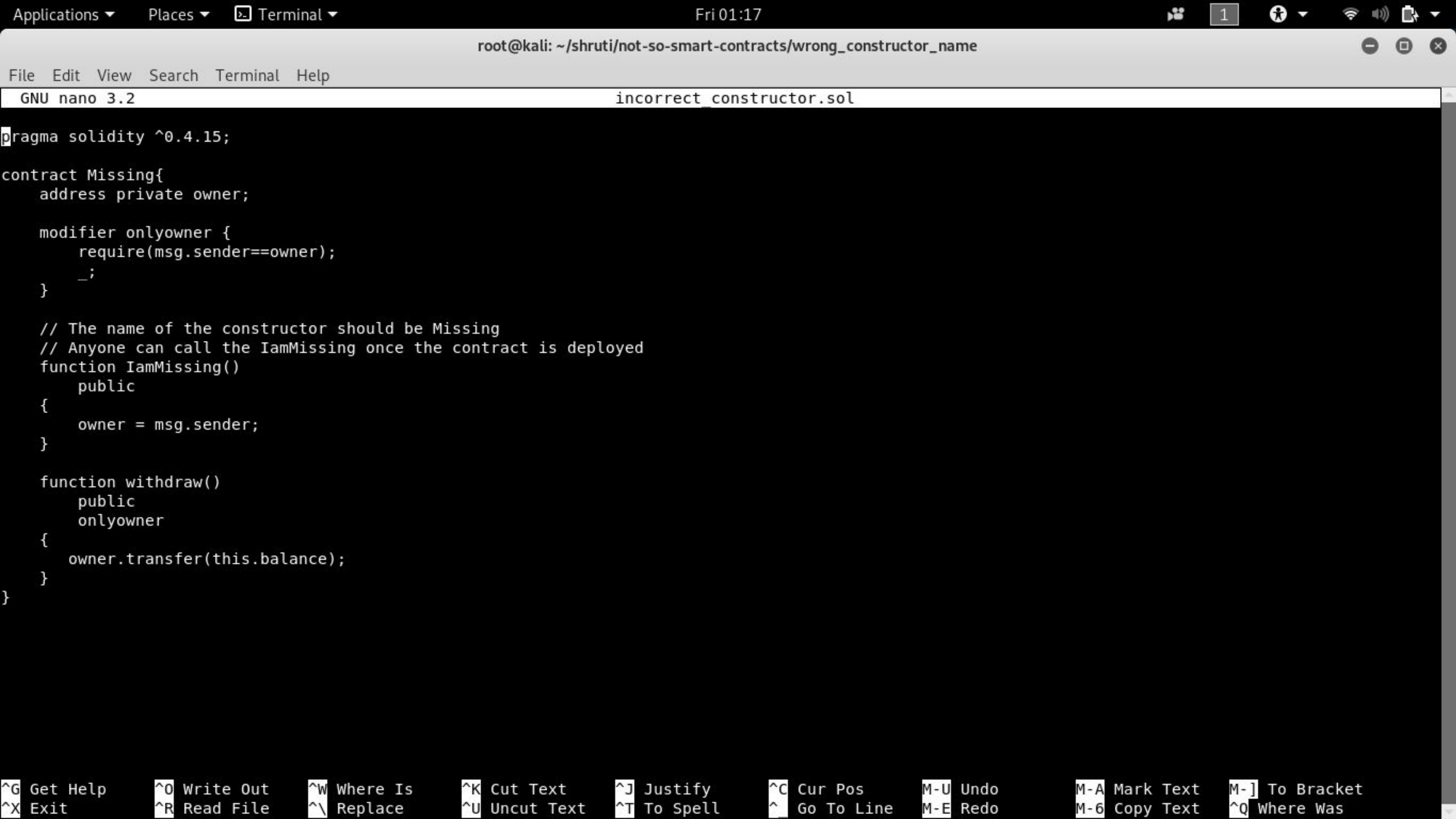
Reentrancy





```
function withdraw(uint _amount) {
        require(balances[msg.sender] >= _amount);
        msg.sender.call.value(_amount)();
        balances[msg.sender] -= _amount;
}
```

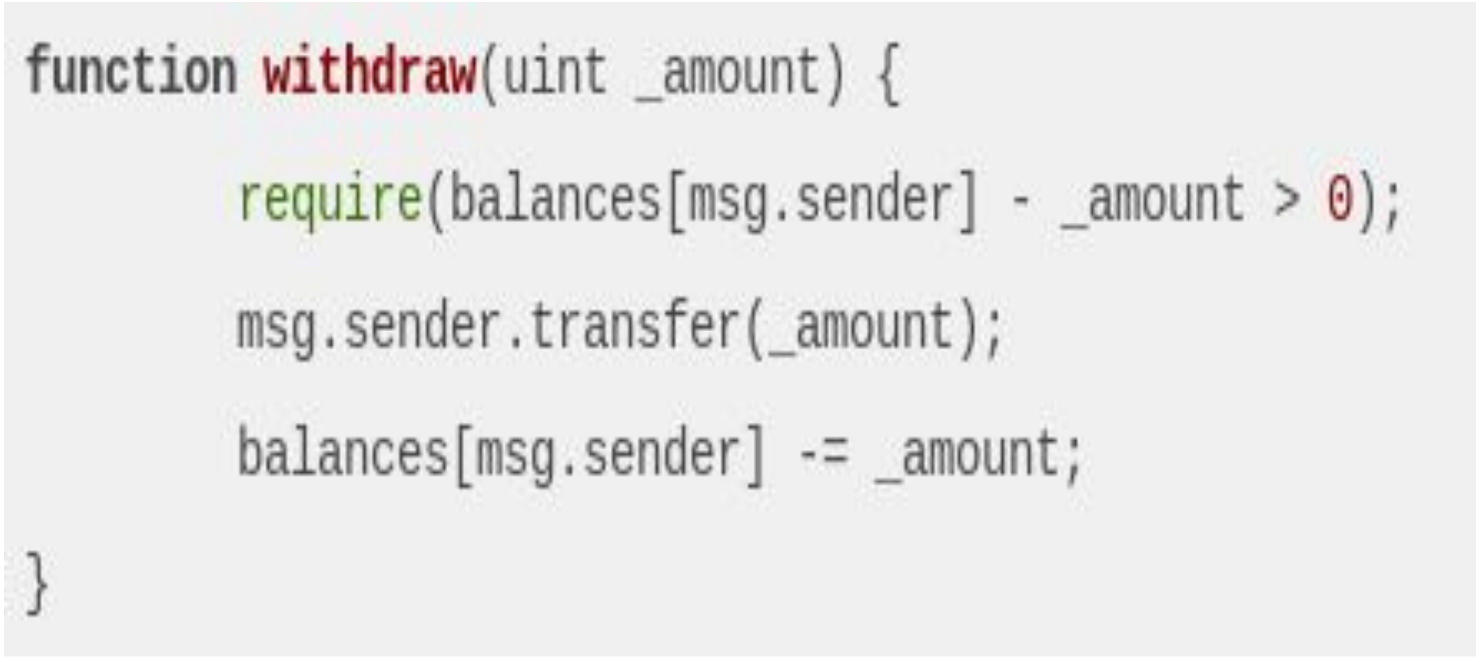
Wrong Constructor


Applications
Places
Terminal
Fri 01:17
root@kali: ~/shruti/not-so-smart-contracts/wrong_constructor_name
File Edit View Search Terminal Help
GNU nano 3.2
incorrect_constructor.sol

```
pragma solidity ^0.4.15;

contract Missing{
    address private owner;

    modifier onlyowner {
        require(msg.sender==owner);
        _;
    }

    // The name of the constructor should be Missing
    // Anyone can call the IamMissing once the contract is deployed
    function IamMissing()
        public
    {
        owner = msg.sender;
    }

    function withdraw()
        public
        onlyowner
    {
       owner.transfer(this.balance);
    }
}
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text M-] To Bracket
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line M-E Redo M-6 Copy Text ^Q Where Was

Arithmetic



```
function withdraw(uint _amount) {
        require(balances[msg.sender] - _amount > 0);
        msg.sender.transfer(_amount);
        balances[msg.sender] -= _amount;
}
```

Denial Of Services

```
function selectNextWinners(uint256 _largestWinner) {
    for(uint256 i = 0; i < largestWinner, i++) {
        // heavy code
    }
    largestWinner = _largestWinner;
}
```

Unusual behaviour

- May or maynot be related to the known bugs
- But worth experimenting if noticed

Any Questions???

References

- https://www.researchgate.net/publication/323545752_Smart_Contracts_Vulnerabilities_A_Call_for_Blockchain_Software_Engineering
- https://www.usenix.org/conference/usenixsecurity18/presentation/krupp
- https://eprint.iacr.org/2016/1007.pdf
- https://dasp.co/
- https://docs.google.com/presentation/d/1qgk82dZ-PJc8CMeZ2Pst_6PFUvKSzdrDe7s-3lEReEg/edit#slide=id.g5040353ef9_0_37
- https://github.com/trailofbits/not-so-smart-contracts

Thank you

https://about.me/shrutirupa/