Smart Contract Security

Readings

- Slides adopted from https://learnblockcha.in/
- Mastering ethereum book.
- Examples

What is smart contract security

- Prevent exploits
- This is difficult for **public blockchains**
  - because they are public
  - code can be read, analysed and called by anybody
- **Ethereum vs Bitcoin ->** EVM simplifies writing and exploiting smart contracts

- Exploits have happened and millions have been “stolen”
- Exploits among even the best teams
- No way to upgrade smart contracts

➡TDD, formal verification, code audits, bug bounty

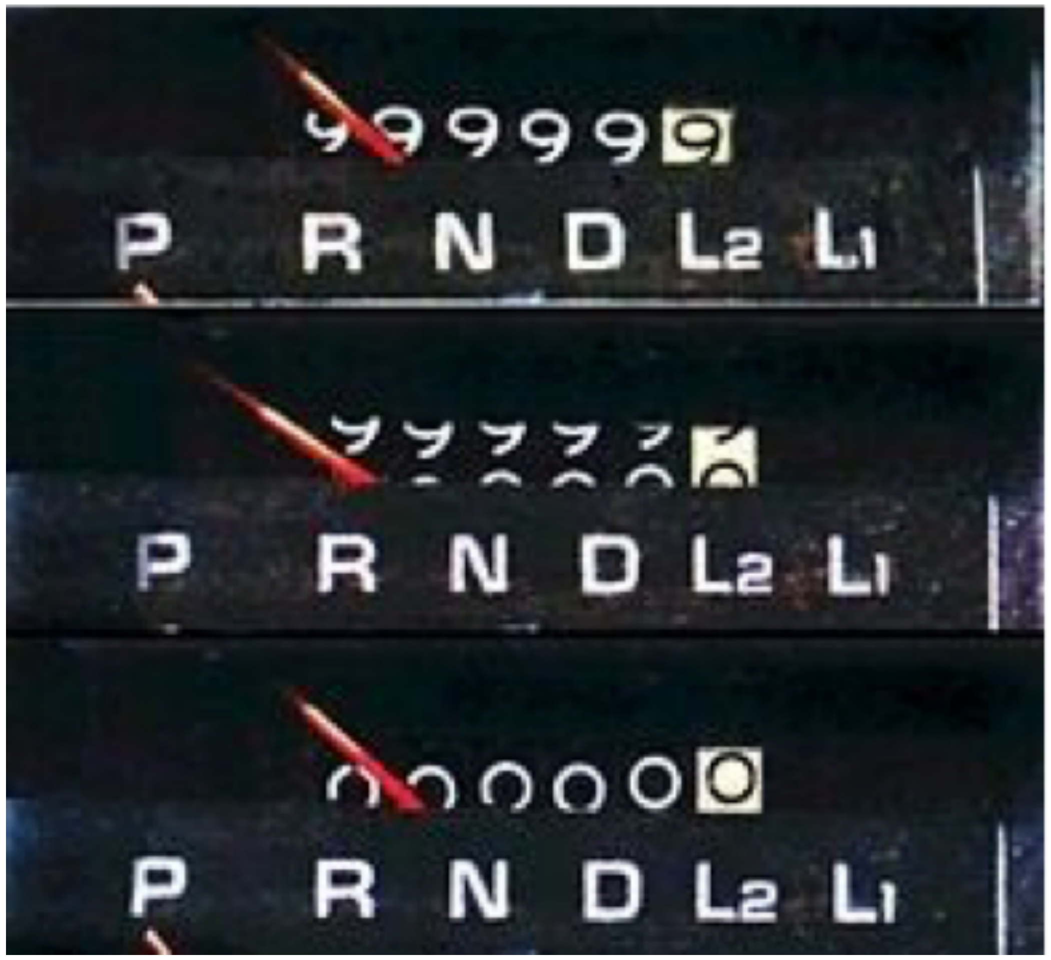
➡Language and platform support

Known vulnerabilities

Integer overflow

Increment a number above its **max value**

- Solidity has max 256 bit number
- **Overflow:** Incrementing 2*256-1 gives 0



After reaching the maximum reading, an odometer or trip meter restarts from zero, called odometer rollover.

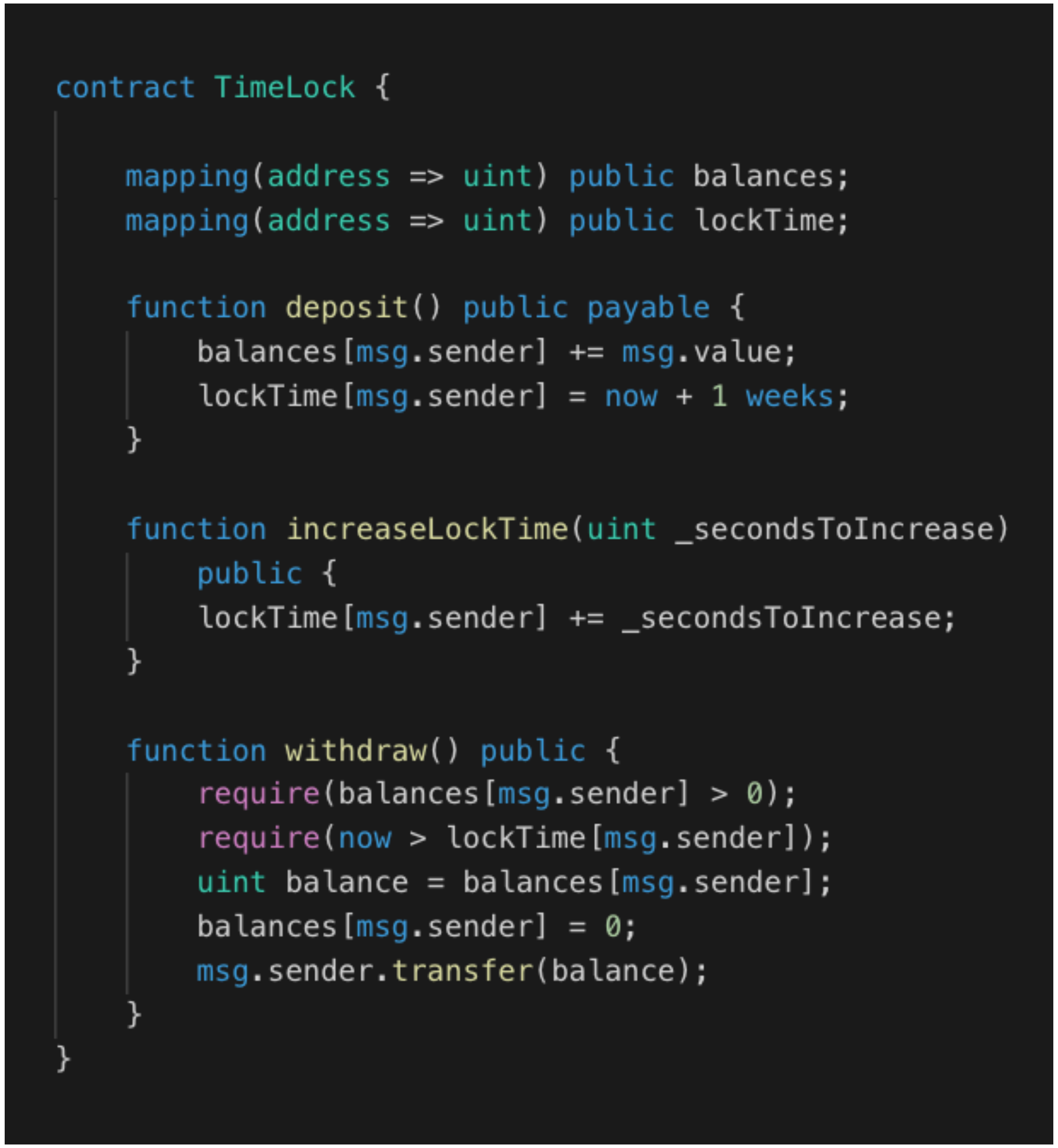
Integer overflows

TimeLock example

- On github
- Call *increaseLockTime* to create overflow and allow immediate withdrawal.

Quiz

- Howto use *require* to guard TimeLock?



```
contract TimeLock {

    mapping(address => uint) public balances;
    mapping(address => uint) public lockTime;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
        lockTime[msg.sender] = now + 1 weeks;
    }

    function increaseLockTime(uint _secondsToIncrease)
        public {
        lockTime[msg.sender] += _secondsToIncrease;
    }

    function withdraw() public {
        require(balances[msg.sender] > 0);
        require(now > lockTime[msg.sender]);
        uint balance = balances[msg.sender];
        balances[msg.sender] = 0;
        msg.sender.transfer(balance);
    }
}
```

Integer overflows

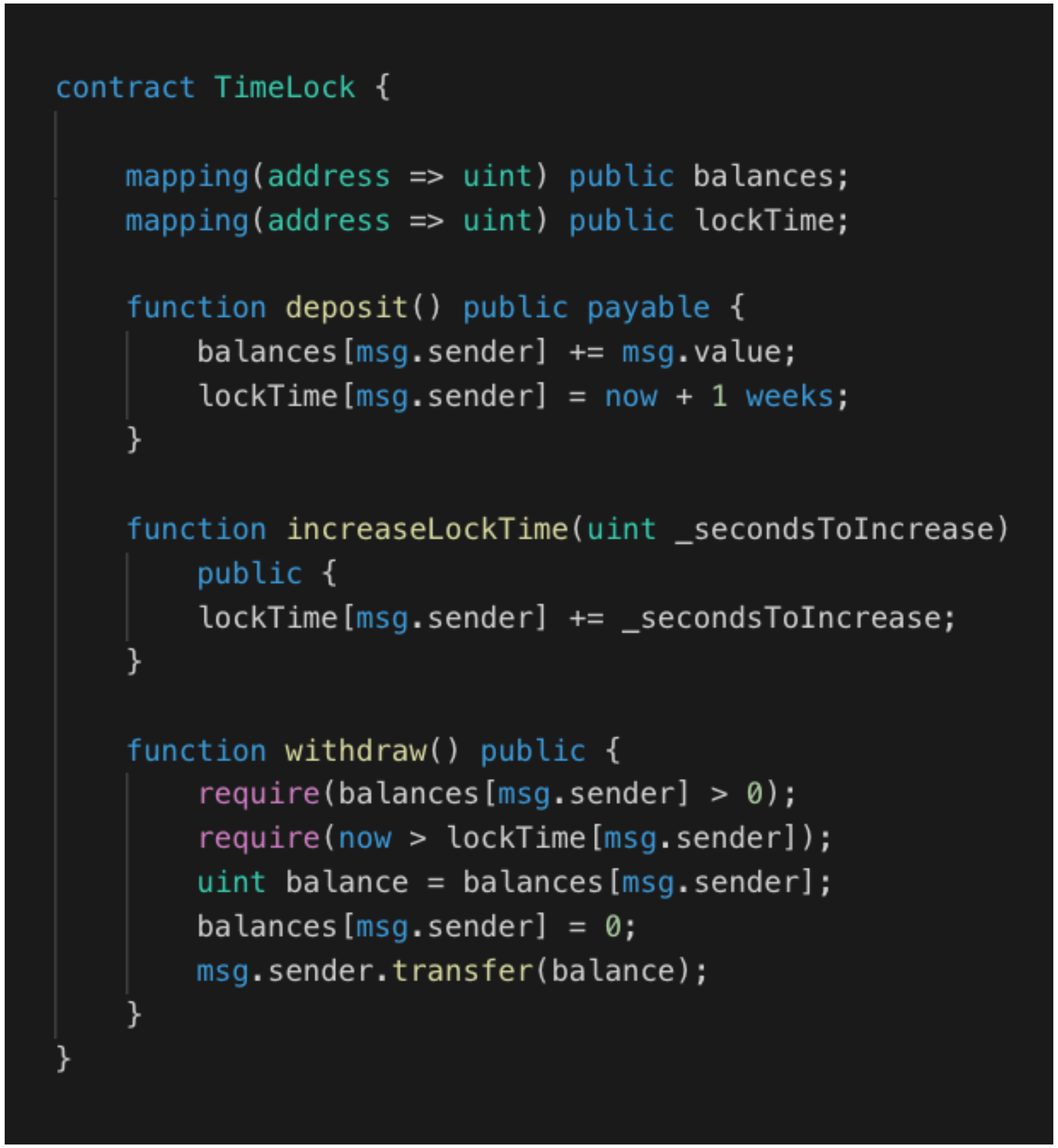
Mitigation

Use require to revert on overflow.

- Use *openzeppelin/safemath*

Quiz

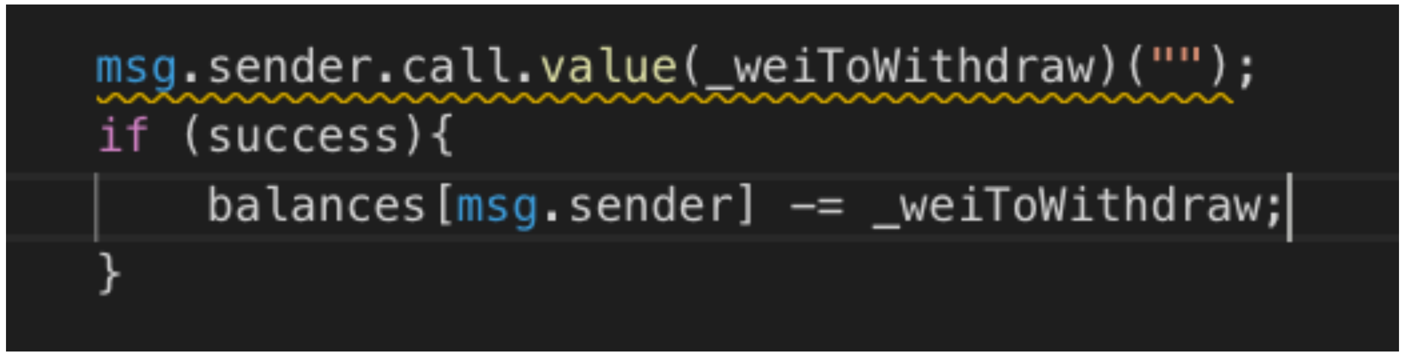
Howto best exploit Token.sol?



```
contract TimeLock {

    mapping(address => uint) public balances;
    mapping(address => uint) public lockTime;

    function deposit() public payable {
        balances[msg.sender] += msg.value;
        lockTime[msg.sender] = now + 1 weeks;
    }

    function increaseLockTime(uint _secondsToIncrease)
        public {
        lockTime[msg.sender] += _secondsToIncrease;
    }

    function withdraw() public {
        require(balances[msg.sender] > 0);
        require(now > lockTime[msg.sender]);
        uint balance = balances[msg.sender];
        balances[msg.sender] = 0;
        msg.sender.transfer(balance);
    }
}
```

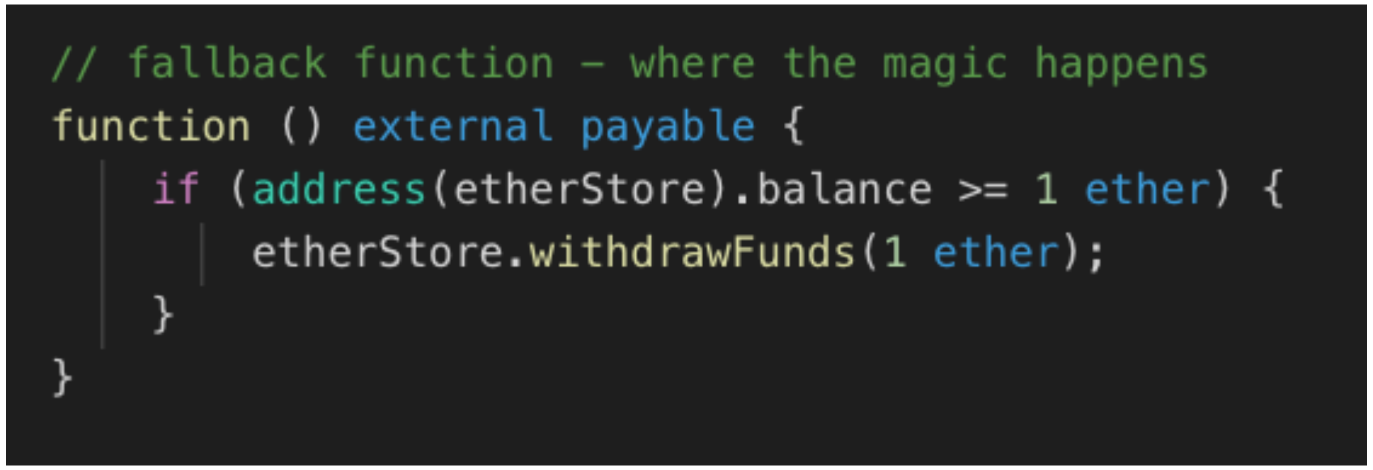
Re-entrancy

Sending money to a contract triggers the fallback function.



```
msg.sender.call.value(_weiToWithdraw)("");
if (success){
    balances[msg.sender] -= _weiToWithdraw;
}
```

Fallback function may recursively re-invoke the current function.



```
// fallback function - where the magic happens
function () external payable {
    if (address(etherStore).balance >= 1 ether) {
        etherStore.withdrawFunds(1 ether);
    }
}
```

Mitigation

- **Pattern**: Reduce balance before sending.
- Use **send** or **transfer**, not call

Method	address.send( )	address.transfer( )	address.call.value( )( )
Possibility to set gas limit	No	No	Yes
Gas limit	2300	2300	Settable
Return value when error	FALSE	Throws exception	FALSE

DOS:

- People will attack your contract, to make it dysfunctional, even if it costs them some money.

Forcing ether

- Ether may be sent to a contract without invoking the fallback function using *selfdestruct*.

Example: EtherGame

Visibility:

- Make helper function private

Randomness:

- Randomness is difficult to get on ethereum

Timestamp

- The small part (e.g. microseconds) of the timestamp can be set arbitrary by the minor

Values are public:

- Values in transactions can be seen by minors and other nodes, before transaction is entered into the block.

Execution order can be influenced:

- By minor that creates a new block
- By other clients (high gas price)

Check library addresses:

- Verify that libraries point to correct addresses, and avoid *delegatecall*.

How would you implement Rock-Paper-Scissors?