

FS19

BLOCKCHAIN, BITCOIN, ETHEREUM III

Ethereum Smart Contracts II

T. Bocek

thomas.bocek@hsr.ch

■ Distributed Systems - In the News

■ Solidity

- Language
- Examples
- ERC20
- Security Considerations
- Ganache/Testonator

Distributed Systems - In the News

■ 05.04.2019: Google Searches for 'Bitcoin' Hit Highest Total Since November

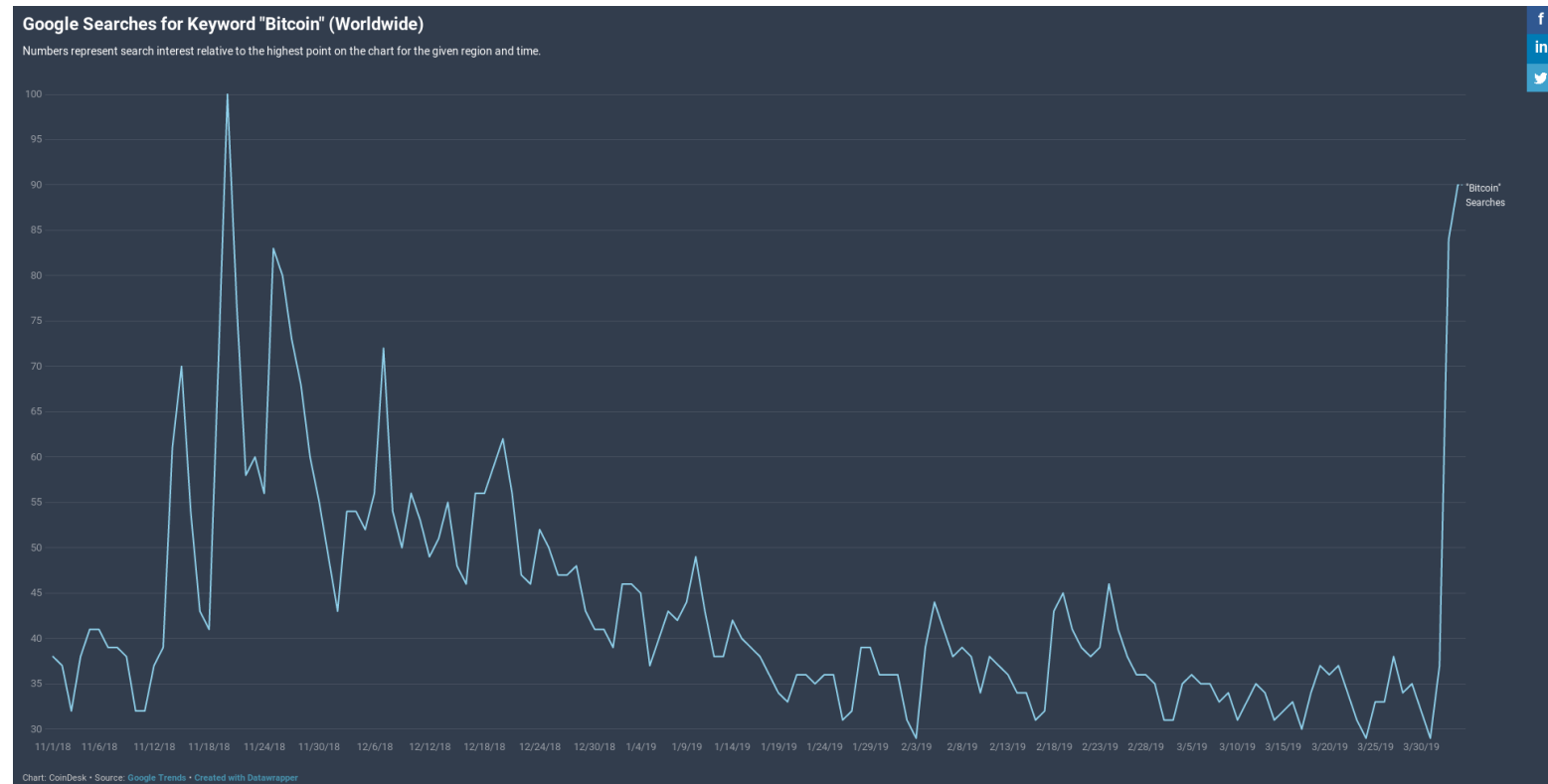
- “Interestingly, most searches came from countries that may not seem to be the most likely of origins.”
- “largest amount of searches on April 2-3 were from St. Helena, Netherlands, South Africa, Nigeria and Austria”

■ 03.04.2019: 5-Star Swiss Hotel Set to Accept Bitcoin Payments

- After Digitec, now Dolder
- Inacta / Bity (provider)

■ 09.04.2019: China says it wants to eliminate bitcoin mining

- Seeking public opinions of industries it wants to encourage, restrict or eliminate
- Mining should be phased out: “not adhere to relevant laws and regulations, were unsafe, wasted resources or polluted the environment”



■ 03.04.2019: Learning to build distributed systems

- Learn through the work of others - Lots of good links:
 - https://www.youtube.com/watch?v=QdzV04T_kec
 - <https://www.the-paper-trail.org/post/2014-08-09-distributed-systems-theory-for-the-distributed-systems-engineer/>
 - <http://book.mixu.net/distsys/>
 - <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.7628>
 - Latency: difference between local and remote is currently between four and five orders of magnitude
 - https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html
 - https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing
- Get hands-on: Challenge Task
- ...
- It takes time

Learning Goals

- I saw the syntax of Solidity and know where to look for it
- I can deploy a Solidity smart contract
- I can write a simple Solidity smart contract
- I know the common pitfalls

■ Types

- bool: true and false, int / **uint** (int8, int16, ..., int256), **address**: 20 bytes, fixed size arrays: bytes1, bytes2, bytes3, ..., bytes32, variable sized: bytes (push), strings

■ Structs

```
struct Account {  
    string name;  
    uint256 amount;  
}
```

■ Mapping

```
mapping (address => uint256) accounts; //mapping with basic types  
mapping (uint256 => Account) accounts; //mapping with structs
```

■ Arrays

```
string[] names
uint256 newLength = names.push("John");
```

■ Another contract

```
pragma solidity ^0.5.7;
contract Example2 {
    struct Account {
        string addr;
        uint256 amount;
    }
    uint256 public counter;
    mapping (uint256 => Account) accounts;
}
```

■ Create getter/setter

```
function get(uint256 nr) public view returns (string) {
    return accounts[nr].addr;
}
function set(uint256 nr, string addr) public {
    require(owner == msg.sender);
    accounts[counter++] = Account(addr, nr);
}
```

■ Read state variables

- Constant function does not modify state
- Reads “for free”

■ Preconditions

■ Special Variables and Functions

```
require(owner == msg.sender);
```

■ Set owner – old syntax

```
pragma solidity ^0.4.0;
contract Example3 {
    address owner;
    function Example3() {
        owner = msg.sender;
    }
}
```

■ Constructor – new syntax

```
constructor(string addr) public {
    owner = msg.sender;
}
```

■ Events – notifications

```
pragma solidity ^0.5.7;
contract Example4 {
    event Message(string msg);
```

■ E.g. in the Geth console

```
at =
browser_example1_sol_example2Contract.at
("0x...")

var event = at.Message(function(error,
result){console.log(result.args.msg)});
```

■ Notary Contract

- Simple Contract
- No constructor

■ Frontend

- Vue.js, dependencies:
 - crypto-js
 - vue
 - web3

■ App.vue, main file

- Template
- Create web3 instance
- Events
 - fileChange()
 - store()

```
pragma solidity ^0.5.7;

contract Notary {

    mapping (address => mapping (bytes32 => uint256)) stamps;

    function store(bytes32 hash) public {
        stamps[msg.sender][hash] = block.timestamp;
    }

    function verify(address recipient, bytes32 hash)
        public view returns (uint256) {
        return stamps[recipient][hash];
    }
}
```

Solidity: Version, Mapping

```
/* Specifies what versions of Solidity a
 * source file can work on
 */
```

```
pragma solidity ^0.5.7;
```

```
contract Notary {
    ...
}
```

```
contract Notary {
    mapping (address => mapping (bytes32 => uint256)) stamps;
    ...
}
```

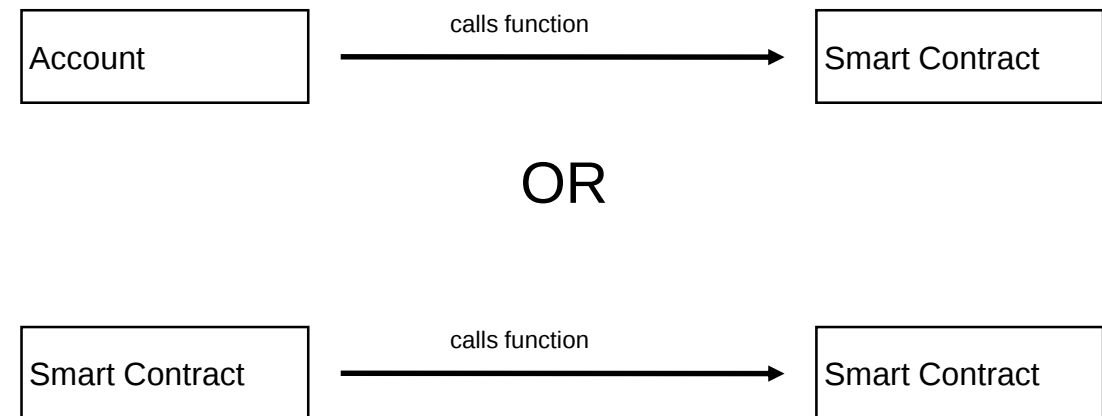
- Ensures that the source code is only meant for compiler versions that are not older than 0.5.0 and will also not work on a compiler starting from version 0.6.0.
- Mapping of address, of mapping of hash, and timestamp

Address	Hash	Timestamp
0xD6df5935cD03a768b7B9E92637A01b25e24cb709	ed52f3833d5d2c920fd43a0972add3555ad149f0201bbd134907498a851a5ec3	2009-06-24 12:39:54 +0900
0xD6df5935cD03a768b7B9E92637A01b25e24cb709	0c911c1d3bde4be991e3d39c2f7e97c621617feaf8f7070e2384c602430ee382	2015-04-14 10:34:24 +0800
0x1530df3e1C69501d4Ecb7E58eB045b90DE158873	0c911c1d3bde4be991e3d39c2f7e97c621617feaf8f7070e2384c602430ee382	2018-07-21 14:33:24 +0900
...	...	...

Solidity: Constructor / Function Calls

```
contract Notary {  
    address owner;  
    /* Initializes the contract, set owner */  
    constructor (uint256 number) public {  
        owner = msg.sender;  
        balances[msg.sender] = number;  
    }  
    ...  
}  
  
contract MyToken {  
    ...  
    /* Send Tokens */  
    function store(bytes32 hash) public {  
        ...  
    }  
}
```

- The constructor is run only once, i.e., when the contract is created!
- Smart contract function can call another smart contract function
- Initiator always needs to be an external account
 - Externally owned account vs. contract account



URL: b.socrative.com/student/

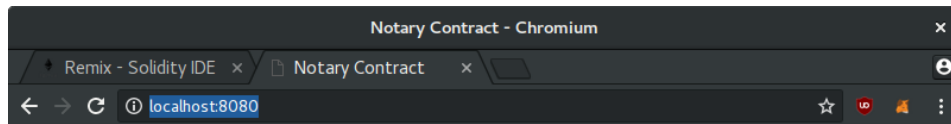
Room: HSR

Run it locally

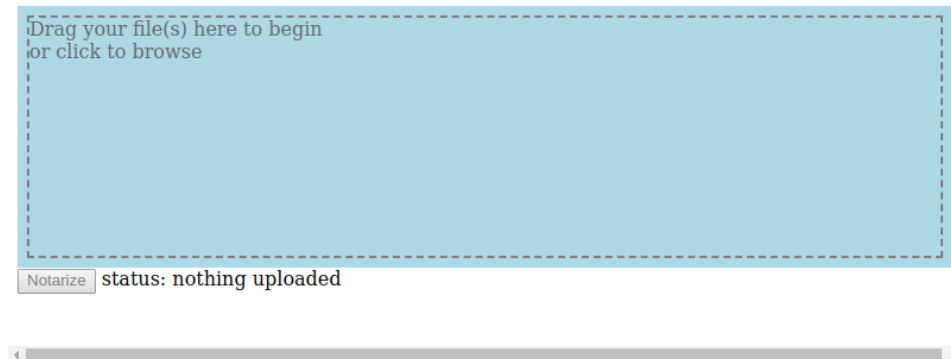
■ Installation

- yarn install
- ./node_modules/.bin/webpack-dev-server

■ Open Browser: <http://localhost:8080/>



Notarize PDF



```
draft@home: ~/git/VSS-web3js
File Edit View Search Terminal Help
draft@home:~/git/VSS-web3js$ ./node_modules/.bin/webpack-dev-server
i [wds]: Project is running at http://localhost:8080/
i [wds]: webpack output is served from /
i [wdm]: Hash: c4c7c0d3279286de6649
Version: webpack 4.7.0
Time: 1139ms
Built at: 2018-05-06 12:57:52
    Asset      Size  Chunks             Chunk Names
main.c4c7c0d3279286de6649.js  947 KiB          0  [emitted]  main
index.html    395 bytes          0  [emitted]
Entrypoint main = main.c4c7c0d3279286de6649.js
[./node_modules/ansi-html/index.js] 4.16 KiB {main} [built]
[./node_modules/loglevel/lib/loglevel.js] 7.68 KiB {main} [built]
[./node_modules/strip-ansi/index.js] 161 bytes {main} [built]
[./node_modules/url/url.js] 22.8 KiB {main} [built]
[./node_modules/vue/dist/vue.esm.js] 286 KiB {main} [built]
[./node_modules/webpack-dev-server/client/index.js?http://localhost:8080] (webpack)-dev-server/client?http://localhost:8080 7.75 KiB {main} [built]
[./node_modules/webpack-dev-server/client/overlay.js] (webpack)-dev-server/client/overlay.js 3.58 KiB {main} [built]
[./node_modules/webpack-dev-server/client/socket.js] (webpack)-dev-server/client/socket.js 1.05 KiB {main} [built]
[./node_modules/webpack/hot sync ^\\.\\.\\log$] (webpack)/hot sync nonrecursive ^\\.\\.\\log$ 170 bytes {main} [built]
[./node_modules/webpack/hot/emitter.js] (webpack)/hot/emitter.js 77 bytes {main} [built]
[./node_modules/webpack/hot/log.js] (webpack)/hot/log.js 1010 bytes {main} [optional] [built]
[./src/App.vue] 908 bytes {main} [built]
[./src/App.vue?vue&type=template&id=7ba5bd90] 194 bytes {main} [built]
[0] multi (webpack)-dev-server/client?http://localhost:8080 ./src 40 bytes {main} [built]
[./src/index.js] 129 bytes {main} [built]
+ 63 hidden modules
Child html-webpack-plugin for "index.html":
  1 asset
  Entrypoint undefined = index.html
[./node_modules/html-webpack-plugin/lib/loader.js!./index.html] 527 bytes {0} [built]
[./node_modules/lodash/lodash.js] 527 KiB {0} [built]
[./node_modules/webpack/buildin/global.js] (webpack)/buildin/global.js 489 bytes {0} [built]
[./node_modules/webpack/buildin/module.js] (webpack)/buildin/module.js 497 bytes {0} [built]
i [wdm]: Compiled successfully.
```

■ MetaMask

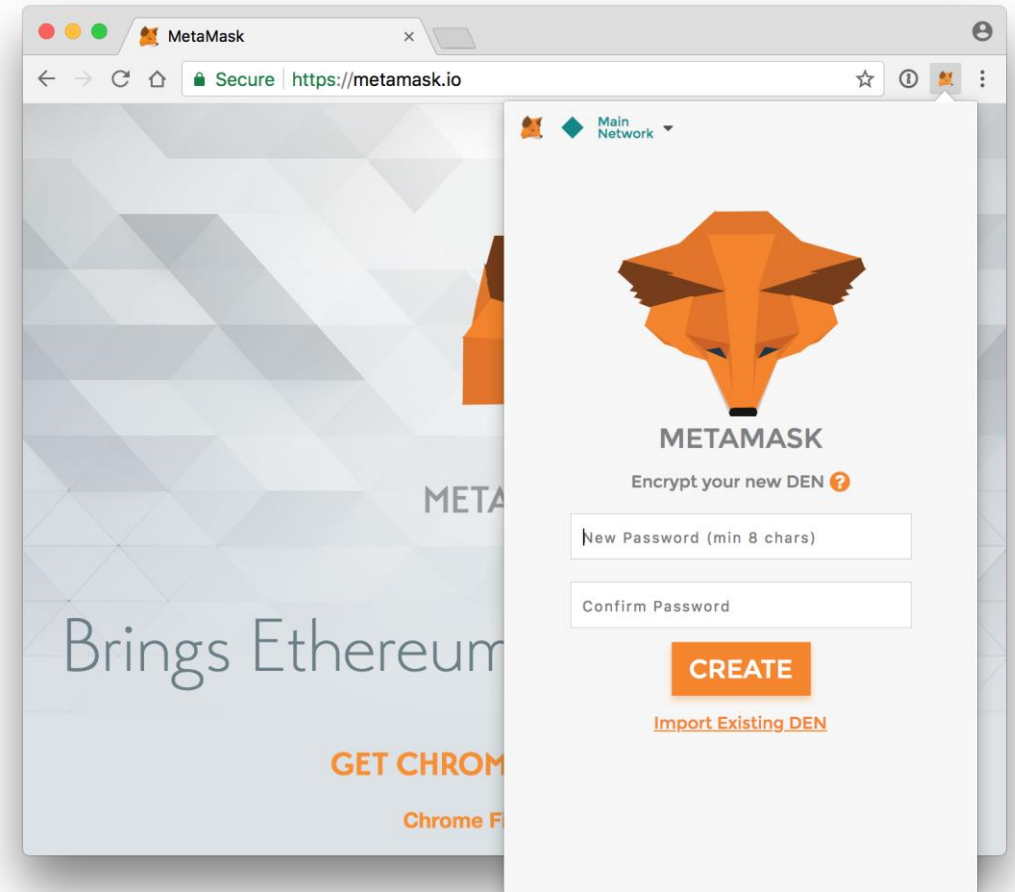
- Browser plugin to make Ethereum transactions in browsers
- Manage your key pairs and sign blockchain transactions
- MetaMask injects javascript library - [web3.js](#)
- Uses [infura](#)

■ Open Remix IDE: <https://remix.ethereum.org>

- Use Notary.sol from <https://github.com/tbocek/VSS-FS18/blob/master/VSS-web3js/Notary.sol>
- Alternatively, use IntelliJ solidity plugin and deploy via geth, parity, or a local test blockchain

■ Compile:

- browser/Notary.sol:8:36:use of "block.timestamp":
"block.timestamp" can be influenced by miners to a certain degree.



Create Contract

■ Connect to MetaMask

- Injected Web3
- If you see your accounts, good!

■ Create contract!

- Add address to the code (manually)

■ Store value

- **0x654cf88c4cff3e62696f7cbb55fbba922b2a75897a010347e371e9398b658c4affa611aa**
- Hash is:
0x4cff3e62696f7cbb55fbba922b2a75897a010347e371e9398b658c4affa611aa
- What is 0x654cf88c?
- **First four bytes of the Keccak (SHA-3) hash of the signature of the function.**
 - Keccak(store(bytes32))

MetaMask Notification

CONFIRM TRANSACTION Rinkeby Test Net

Account 2
25D963...A630
46.661 ETH
36089.67 USD

New Contract

Amount 0 ETH 0.00 USD

Gas Limit UNITS

Gas Price GWEI

Max Transaction Fee 0.000353 ETH 0.27 USD

Max Total 0.000353 ETH 0.27 USD

Data included: 500 bytes

RESET SUBMIT REJECT

Environment Rinkeby (4)

Account

Gas limit

Value wei

Notary

Create

Load contract from Address At Address

0 pending transactions

0 contract instances

URL: b.socrative.com/student/

Room: HSR

■ ERC20 is a technical standard for smart contracts on the Ethereum blockchain for implementing tokens

- proposed on November 19, 2015
- January 2018: more than [21000](#) [ERC20 token](#) contracts: EOS, Tronix, VeChain, OmiseGO, MOD Token

■ ERC20 Token (Simplified)

- No allowance / approval / transferFrom, also no Approval event
- Using SafeMath (always use it)
- Creator gets 1000 coins

```
contract SimpleERC20 {  
    function totalSupply() public view returns (uint256);  
    function balanceOf(address who) public view returns  
(uint256);  
    function transfer(address to, uint256 value) public  
returns (bool);  
    event Transfer(address indexed from, address indexed to,  
uint256 value);  
}
```

```
string public constant name = "VSS-TOKEN";  
string public constant symbol = "VST";  
uint8 public constant decimals = 18;
```

```
using SafeMath for uint256;  
mapping(address => uint256) balances;  
uint256 totalSupply_;
```

```
constructor() public {  
    totalSupply_ = 1000 * (10**18);  
    balances[msg.sender] = totalSupply_;
```

```
17 }
```

...

■ Transfer ownership

- Important for minting: after minting, set new owner so that the private key is never exposed
- Private key never on the minting machine

■ Token lockups

- Vest tokens – make sure big investor don't dump
- Team vesting – show the world that you believe in your token

■ Minting

- Lock contract during minting

```
function transferOwnership(address _newOwner)
public {
    require(owner == msg.sender);
    owner = _newOwner;
}
```

```
mapping(address => uint256) lockups;
...
if (lockups[msg.sender] != 0) {
    require(now >= lockups[msg.sender]);
}
```

```
bool public mintingDone = false;
...
require(mintingDone == true);
```

■ Utility Token – ERC [223](#) / [677](#)

- Eliminates the problem of lost tokens
 - QTUM, \$1,204,273 lost
 - EOS, \$1,015,131 lost
- Allows developers to handle incoming token transactions
- Backwards [compatible](#)?
 - ERC 223: throw if transferring to a contract that does not implement tokenFallback... (corner cases)
- ERC 677 fully backwards compatible, but tokens still can be lost

■ transferAndCall()

- Implement token now, define utility later

```
contract ERC677 {
    event Transfer(address indexed _from, address indexed _to, uint256 _value,
bytes _data);
    function transferAndCall(address _to, uint _value, bytes _data) public
returns (bool success);
}

contract ERC677Receiver {
    function tokenFallback(address _from, uint _value, bytes _data) public;
}

// ERC677 functionality
function transferAndCall(address _to, uint _value, bytes _data) public returns
(bool) {
    require(transfer(_to, _value));
    ...
    if (isContract(_to)) {
        ERC677Receiver receiver = ERC677Receiver(_to);
        receiver.tokenFallback(msg.sender, _value, _data);
    }
}
//ERC223
function transfer(address _to, uint _value) public returns (bool success) {
    bytes memory empty;
    if(isContract(_to)) {
        return transferToContract(_to, _value, empty);
    }
    else {
        return transferToAddress(_to, _value, empty);
    }
} 19
}
```

URL: b.socrative.com/student/

Room: HSR

Security Considerations

■ List of the top smart contract vulnerabilities

1. Reentrancy (DAO 3.5M, 50M)
2. Access Control (Parity 150K, 30M)
3. Arithmetic Issues (DAO)

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

Contract AA {
    function attack() {
        A a = A(addressOfA);
        a.withdraw(100);
    }
}

function () payable {
    A a = A(addressOfA);
    a.withdraw(100);
}

function initContract() public {
    owner = msg.sender;
}

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

■ List of the top smart contract vulnerabilities

1. Reentrancy (DAO 3.5M, 50M)
2. Access Control (Parity 150K, 30M)
3. Arithmetic Issues (DAO)
4. Unchecked Return Values For Low Level Calls
5. Denial of Service (Parity 500K, 300M)
6. Bad Randomness (400K)
 - Future blockhash as random source, but miner can influence it.

```
function withdraw(uint256 _amount) public {
    require(balances[msg.sender] >= _amount);
    balances[msg.sender] -= _amount;
    etherLeft -= _amount;
    msg.sender.send(_amount);
}
```

```
function getEtherBalanceIndex(address _address) internal
returns (uint256) {
    for (uint256 i = 0; i < investors.length; i++) {
        if (investors[i] == _address) {
            return i;
        }
    }
    return investors.length;
}
```

```
function play() public payable {
    require(msg.value >= 1 ether);
    if (block.blockhash(blockNumber) % 2 == 0) {
        msg.sender.transfer(this.balance);
    }
}
```

■ List of the top smart contract vulnerabilities

1. Reentrancy (DAO 3.5M, 50M)
2. Access Control (Parity 150K, 30M)
3. Arithmetic Issues (DAO)
4. Unchecked Return Values For Low Level Calls
5. Denial of Service (Parity 500K, 300M)
6. Bad Randomness (400K)
7. Front-Running
8. Time Manipulation - A malicious miner sets own timestamp ~ [900s](#)
9. Short Address Attack – not yet exploited
10. Unknown Unknowns

1. A smart contract publishes an RSA number ($N = \text{prime1} \times \text{prime2}$).
2. A call to its `submitSolution()` public function with the right `prime1` and `prime2` rewards the caller.
3. Alice successfully factors the RSA number and submits a solution.
4. Someone on the network sees Alice's transaction (containing the solution) waiting to be mined and submits it with a higher gas price.
5. The second transaction gets picked up first by miners due to the higher paid fee. The attacker wins the prize.

We believe more security audits or more tests would have made no difference. The main problem was that reviewers did not know what to look for.

Christoph Jentzsch

■ Multiple Exchanges Suspend ERC20 Token Trading Due To Potential BatchOverflow Bug

- Integer overflows in multiple contracts
- Poloniex suspended all ERC20 token deposits and withdrawals,
- OKEX's decision to halt ERC20 deposits

```
255 function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {  
256     uint cnt = _receivers.length;  
257     uint256 amount = uint256(cnt) * _value;  
258     require(cnt > 0 && cnt <= 20);  
259     require(_value > 0 && balances[msg.sender] >= amount);  
260  
261     balances[msg.sender] = balances[msg.sender].sub(amount);  
262     for (uint i = 0; i < cnt; i++) {  
263         balances[_receivers[i]] = balances[_receivers[i]].add(_value);  
264         Transfer(msg.sender, _receivers[i], _value);  
265     }  
266     return true;  
267 }  
268 }
```

URL: b.socrative.com/student/

Room: HSR

■ Minting done in batches

- Around 100/150 until max gas used

■ Only owner can mint

- Once minting finished no one can ever mint

■ Check maxSupply

- Don't mint more tokens, check important, as minter needs to respect limits

■ Emit from:0

- Discussion if from is 0 or contract address

```
function mint(address[] _recipients, uint256[] _amounts)
public {
    require(owner == msg.sender);
    require(mintingDone == false);
    require(_recipients.length == _amounts.length);
    require(_recipients.length <= 256);

    for (uint8 i = 0; i < _recipients.length; i++) {
        address recipient = _recipients[i];
        uint256 amount = _amounts[i];

        // enforce maximum token supply
        require(totalSupply_.add(amount) <= maxSupply);

        balances[recipient] =
balances[recipient].add(amount);
        totalSupply_ = totalSupply_.add(amount);

        emit Transfer(0, recipient, amount);
    }
}
```

■ ERC865: Pay transfers in tokens instead of gas, in one transaction

- Delegate transfer of tokens to a third party
- UX: pay service in tokens – user needs to have ETH **and** tokens
 - Create account at exchange, KYC, Bank transfer – and wait

■ Off-chain service to get signature from 3rd party service

■ Use signature to check if transfer tokens allowed

- Signature malleability - OpenZeppelin

■ But, ERC677 – add delegatedTransferAndCall()

```
function delegatedTransfer(
    uint256 _nonce,
    address _from,
    address _to,
    uint256 _value,
    uint256 _fee,
    uint8 _v,
    bytes32 _r,
    bytes32 _s) public returns (bool) {

    uint256 total = _value.add(_fee);
    require(_from != address(0));
    require(_to != address(0));
    require(total <= balances[_from]);
    require(_nonce > nonces[_from]);

    address delegate = msg.sender;
    address token = address(this);
    bytes32 delegatedTxnHash = keccak256(delegate, token, _nonce, _from, _to,
    _value, _fee);
    address signatory = ecrecover(delegatedTxnHash, _v, _r, _s);
    require(signatory == _from);

    balances[_from] = balances[_from].sub(total);
    balances[_to] = balances[_to].add(_value);
    balances[delegate] = balances[delegate].add(_fee);
    nonces[_from] = _nonce;

    DelegatedTransfer(_from, _to, delegate, value, fee);
    return true;
}
```

Random Numbers

■ Random Numbers

- There is no random number in Ethereum
- Every EVM needs to come to the same result

■ Random Numbers from Oracle (External source)

- Or: commitment schemes

■ Or: use future blockhash

- Only up to 256 blockhashes from the past can be accessed.
- Deduct / add tokens/ETH from the past address
- Miner can influence the random value in a sense
 - Value needs to be low (miner gets 2 ETH)

```
function transfer(address _to, uint256 _value) public returns (bool) {
    ...
    //since this is a lucky coin, the value transferred is not what you
    expect
    luckyTransfer();
    previousTransferBlockNr = block.number;
    previousTransferAddress = msg.sender;
    ...
    uint256 val = _value.sub(potIncrease);
    pot = pot.add(potIncrease);
}

function luckyTransfer() private {
    if(block.number != previousTransferBlockNr && (block.number -
previousTransferBlockNr) < 256 ) {
        uint256 rnd =
uint256(keccak256(block.blockhash(previousTransferBlockNr)));
        if(rnd % 200 == 0) { //5%
            balances[previousTransferAddress] =
balances[previousTransferAddress].add(pot);
            Transfer(this, previousTransferAddress, pot);
            //tokens are from pot, thus no tokens are created from thin air
            pot = 0;
            previousTransferBlockNr = 0;
            previousTransferAddress = 0;
        }
    }
}
```

- «Ganache» for Java
- Ethereum Test Blockchain
 - 10 Accounts with 10 Ethers
- Uses [web3j](#)
 - But **no** contract compilation from Solidity to Java
- Uses [ethereumJ](#), [solcJ](#), [jsonrpc](#), and [jetty](#)
 - Wires them up

```
plugins {  
    id 'java'  
}  
  
group 'ch.hsr.dsl'  
archivesBaseName = 'uzh-sc'  
version '1.0-SNAPSHOT'  
  
repositories {  
    mavenLocal()  
    mavenCentral()  
    maven { url 'https://dl.bintray.com/ethereum/maven/' }  
}  
  
dependencies {  
    testCompile 'io.iconator:testonator:1.0.35'  
    testCompile 'junit:junit:4.12'  
}
```

■ Goal: make testing easy

- Easy API
- `blockchain.call(dcDOS, "removeWhitelist", dcTest.contractAddress());`
 - Call `removeWhitelist` with a parameter on the `dcDOS` contract
 - Or use `FunctionBuilder` to be more explicit. When contract and function known, type conversion can be done
 - If calling external function, `FunctionBuilder/Fb` needs to be used
- Contracts compiled within Testonator:
 - `Map<String, Contract> cnt = TestBlockchain.compile(File or String)`
- Deploy contract
 - `dcDOS = TestBlockchain.deploy(CREDENTIAL_0, cnt.get("Exempl2"));`
- `testBlockchain.callConstant` can return values
- `testBlockchain.call` never returns values, only returns events (limitation of web3)

▼ ✓ <default package>	10 s 147 ms
▼ ✓ TestERC20	7 s 487 ms
✓ testApproveWithdrawTwice	912 ms
✓ testWithdrawTransferFrom	427 ms
✓ testEventMintAll	189 ms
✓ testEventMinting	171 ms
✓ testApprove	330 ms
✓ testTransferFail	292 ms
✓ testTransferOwnership	388 ms
✓ testTransferFailContract	278 ms
✓ testWithdraw	352 ms
✓ testApproveMax	301 ms
✓ testApproveWithdrawNoAllowance	235 ms
✓ testApproveWithdraw	324 ms
✓ testApproveWithdrawTransfer	352 ms
✓ testReverse	147 ms
✓ testEventMintAllFail	141 ms
✓ testEventApprove	263 ms
✓ testEventTransfer	288 ms
✓ testLockup	286 ms
✓ testEventTransferFrom	267 ms
✓ testEventTransferZero	238 ms
✓ testTransferFailFlag	203 ms
✓ testTransferFailZero	276 ms
✓ testApproveFailZero	255 ms
✓ testTransfer	257 ms

- **src/test/java**
- **Run local node accessible via Jetty, can connect to geth/parity**
- **Compile contracts, result is stored in a map**
- **After each test, reset, after all tests shutdown Jetty**
- **Deploy with “deploy”**
- **Call with “call”, returns events emitted, not return values**
- **Call constant with “callConstant”, returns return values**

```
@BeforeClass
public static void setup() throws Exception {
    blockchain = TestBlockchain.runLocal();
    contracts = compile(contractFile1);
}

@AfterClass
public static void tearDown() throws Exception {
    blockchain.shutdown();
}

@After
public void afterTests() {
    blockchain.reset();
}
```

```
DeployedContract deployed = blockchain.deploy(CREDENTIAL_0,
contracts.get("DOS"));
```

```
List<Event> events = blockchain.call(deployed, "mint", ...);
```

```
List<Type> result =blockchain.callConstant(deployed,
“balanceOf”, ...);
```

URL: b.socrative.com/student/

Room: HSR

Questions?

Prof. Dr. Thomas Bocek
thomas.bocek@hsr.ch



Upvote this lecture 😊

1MGDdtoJHQkDEmPwWtwEQxhtRvJfMMAz83