



Secure Token Development and Deployment

Dmitry Khovratovich and Mikhail Vladimirov,

[University of Luxembourg and ABDK Consulting](#)

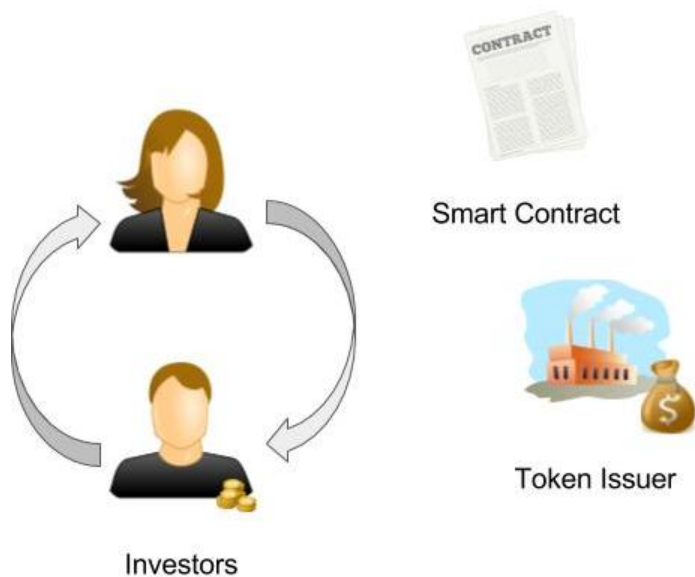
ERC-20 tokens and ICO

- ERC-20 standard: developed in late 2015, de-facto standard for fungible assets.
- Many enterprises decided to raise funds via Initial Coin Offering (ICO):
 - Users deposit money to a dedicated web-service or Ether to a bookbuilding contract;
 - The total investment is converted to a fixed number of tokens;
 - A smart contract is issued, where each user gets tokens according to his investment share.
- In 2016, 64 ICOs gathered over \$103 mln. (*CoinDesk*)



..iCONOMI

ERC-20 functions



Function	Description
<code>totalSupply()</code>	Total number of tokens in circulation
<code>balanceOf(A)</code>	Number of tokens belonging to A
<code>transfer(A,x)</code>	Send x tokens to A
<code>transferFrom(A,x)</code>	Withdraw x tokens from A
<code>approve(A,x)</code>	Approve A to withdraw tokens from me
<code>allowance(A,B)</code>	How much B can withdraw from A

Security

- Security of the ICO process and contract maintenance is a vital issue.
- Threats:
 - Token loss due to contract misbehavior;
 - Availability loss: external contracts (say, exchanges) can not interact with the token contract due to its errors.
- Vulnerabilities:
 - Insecure code;
 - Error-prone interface.



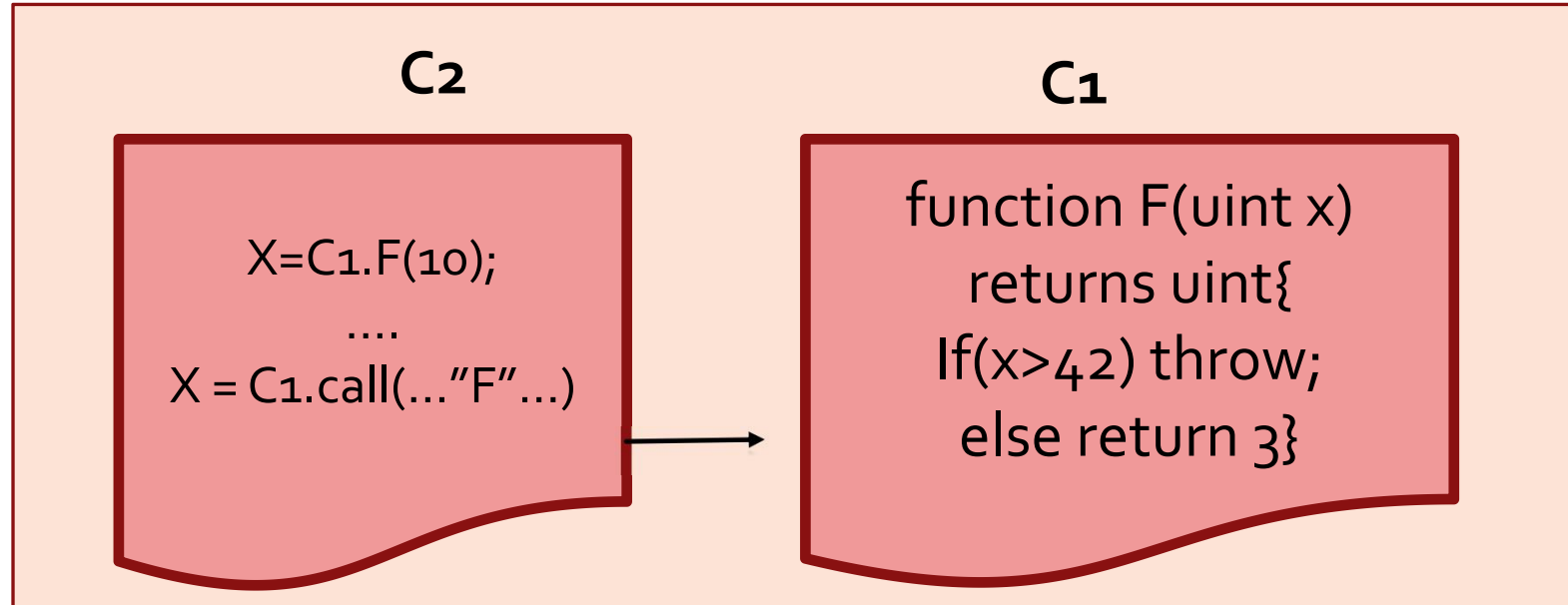
ERC-20 functions

- `transfer(address _to, uint256 _value)` returns (bool success) -
 - Send `_value` amount of tokens to address `_to`
- `balanceOf (address _owner)` constant returns (uint256 balance) -
 - number of tokens belonging to address
- `totalSupply()` constant returns (uint256 totalSupply) -
 - total number of tokens
- `approve(address _spender, uint256 _value)` returns (bool success) -
 - approve the spender to withdraw tokens from my account, multiple times;
- `allowance(address _owner, address _spender)` constant returns (uint256 remaining)
 - how much the spender can withdraw
- `transferFrom(address _from, address _to, uint256 _value)` returns (bool success) - transfer tokens from t



Current Insecurity: Implementations

Exceptions in Solidity



- C1.F() can not catch exceptions;
- C1.call() can not get returned values.
- Exceptions must be for exceptional events!

Exceptions instead of value return

Code example:

```
function transferFrom(address _from, address _to, uint _value) returns (bool success) {
    var _allowance = allowed[_from][msg.sender];

    balances[_to] = safeAdd(balances[_to], _value);
    balances[_from] = safeSub(balances[_from], _value);
    allowed[_from][msg.sender] = safeSub(_allowance, _value);
    Transfer(_from, _to, _value);
    return true;
}

function safeSub(uint a, uint b) internal returns (uint) {
    assert(b <= a);
    return a - b;
}
```


Exceptions instead of value return

Code example:

```
function transferFrom(address _from, address _to, uint _value)
returns (bool success) {
    var _allowance = allowed[_from][msg.sender];

    balances[_to] = safeAdd(balances[_to], _value);
    balances[_from] = safeSub(balances[_from], _value);
    allowed[_from][msg.sender] = safeSub(_allowance, _value);
    Transfer(_from, _to, _value);
    return true;
}
```

- If the _from balance is insufficient, or allowance is too low, exception is thrown (must return false by the standard).

Relying on future exceptions

```
• function transferFrom(address _from, address _to, uint _value)  
  returns (bool success) {  
    var _allowance = allowed[_from][msg.sender];  
    balances[_to] = safeAdd(balances[_to], _value);  
    balances[_from] = safeSub(balances[_from], _value);  
    allowed[_from][msg.sender] = safeSub(_allowance, _value);  
    Transfer(_from, _to, _value);  
    return true;  
  }
```

- The `_to` balance is always increased.
- In order to revert the operation, the allowance decrease **MUST** throw exception.

Overflows in SafeMath

```
• function safeMul(uint a, uint b) internal returns  
(uint) {  
    uint c = a * b;  
    assert(a == 0 || c / a == b);  
    return c;  
}
```

- Code relies on the (undocumented) behavior of overflow in Solidity.
- First creates overflow, then tries to detect it.
- The overflow might be treated differently in the future, so potential compatibility loss.
- Same with safeAdd.



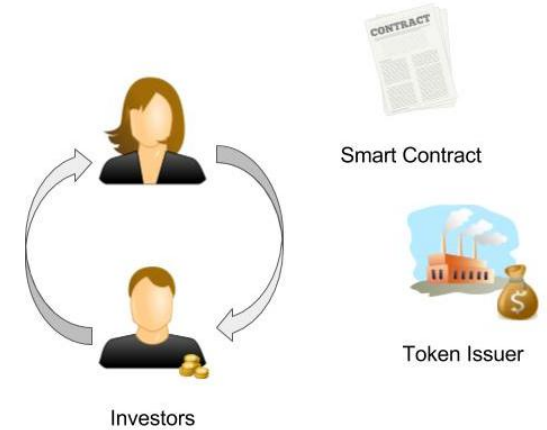
Current Insecurity: the Approve() function

Attack on the ERC-20 standard

Approve

- `approve(address _spender, uint256 _value)`
 _spender can withdraw from your account up to _value tokens.
- Subsequent `approve()` *overrides* the previous one.
- `transferFrom(address _from, address _to, uint256 _value)`
 spender withdraws _value tokens from _from and sends them to _to.
- `transferFrom` logs a Transfer event where the spender is not listed(!).

Attack



1. Alice approves Bob with N tokens;
 2. Alice decides to change Bob's allowance and approves him with M tokens;
 3. Bob notices the last transaction before it is mined and withdraws N tokens from Alice sending them to Carol.
 4. After Alice's transaction is mined, Bob withdraws M more tokens from Alice sending them to Carol, thus taking $M+N$ in total.
- Bob's actions are not properly logged: the Transfer event gives the addresses of Alice and Carol, not Bob.
 - Even more confusion if Carol has allowance from Alice as well.

Our suggestion: atomic approval and new events

1) **function** approve(**address** _spender, **uint256** _currentValue, **uint256** _value) **returns** (**bool** success)

If current allowance for _spender is equal to _currentValue, then overwrite it with _value and return true, otherwise return false.

2) **event** Transfer(**address indexed** _spender, **address indexed** _from, **address indexed** _to, **uint256** _value)

3) **event** Approval(**address indexed** _owner, **address indexed** _spender, **uint256** _oldValue, **uint256** _value)



Our approach



Clear requirements and behavior

- Define functional requirements for the token contract and API;
- Describe all use cases;
- State security claims explicitly.

Detailed use cases

2.4. ERC20:TransferFrom

Actors: Spender, Smart Contract

Goal: Spender wants to transfer certain number of tokens from the owner of certain source address to the owner of certain destination address

Main Flow:

1. Spender calls method on Smart Contract providing the following information as method parameters: number of tokens to transfer, source address, destination address
2. Transfers are not currently frozen
3. Spender is currently allowed to transfer requested number of tokens belonging to the owner of source address
4. The owner of the source address has enough tokens to transfer
5. Smart Contract transfers requested number of tokens from the owner of source address to the source address to the owner of the destination address
6. Smart Contract reduces number of tokens belonging to the owner of source address that Spender is allowed to transfer
7. Some tokens actually did change hands during transfer, i.e. number of tokens transferred is more than zero and destination address is not the same as source address
8. Smart Contract logs token transfer event with the following information: number of tokens transferred, source address, destination address
9. Smart Contract returns success indicator to Spender

Exceptional Flow 1:

1. Same as in main flow
2. Transfers are currently frozen
3. Smart Contract returns error indicator to Spender

Exceptional Flow 2:

1. Same as in main flow
2. Same as in main flow
3. Spender is currently not allowed to transfer requested number of tokens belonging to the owner of source address
4. Smart Contract returns error indicator to Spender

Exceptional Flow 3:

1. Same as in main flow
2. Same as in main flow
3. Same as in main flow
4. The owner of the source address does not have enough tokens to transfer
5. Smart Contract returns error indicator to Spender

Exceptional Flow 4:

1. Same as in main flow
2. Same as in main flow
3. Same as in main flow
4. Same as in main flow
5. Same as in main flow
6. Same as in main flow
7. No tokens actually did change hands during transfer, i.e. number of tokens transferred is zero or destination address is not the same as source address
8. Smart Contract returns success indicator to Spender


Elaborate tests

```
{ name: "Carol tries to transfer 100 Dave's tokens to himself while Dave does not have any tokens",
  body: function (test) {
    assert (
      'test.standardTokenWrapper.allowance (test.dave.address, test.carol.address) == 1000',
      test.standardTokenWrapper.allowance (test.dave.address, test.carol.address) == 1000);
    assert ( 'test.standardTokenWrapper.balanceOf (test.dave.address) == 0',
      test.standardTokenWrapper.balanceOf (test.dave.address) == 0);
    personal.unlockAccount (test.alice, "");
    test.tx = test.carol.execute (
      test.standardTokenWrapper.address,
      test.standardTokenWrapper.transferFrom.getData (
        test.dave.address, test.dave.address, 100, 0,
        {from: test.alice, gas: 1000000});
  }, { name: "Make sure transfer failed",
  precondition: function (test) {
    miner.start ();
    return web3.eth.getTransactionReceipt (test.tx); },
  body: function (test) {
    miner.stop ();
    var transferEvents = test.standardTokenWrapper.Transfer (
      {}, {
        fromBlock: web3.eth.getTransactionReceipt (test.tx).blockNumber,
        toBlock: web3.eth.getTransactionReceipt (test.tx).blockNumber
      }).get ();
    assert (
      'transferEvents.length == 0',
      transferEvents.length == 0);
```

```
var execResultEvents = test.carol.Result (
  {},
  {
    fromBlock: web3.eth.getTransactionReceipt (test.tx).blockNumber,
    toBlock: web3.eth.getTransactionReceipt (test.tx).blockNumber
  }).get ();
assert (
  'execResultEvents.length == 1', execResultEvents.length == 1);
assert (
  'execResultEvents [0].args._value',
  execResultEvents [0].args._value);
var resultEvents = test.standardTokenWrapper.Result (
  {},
  {
    fromBlock: web3.eth.getTransactionReceipt (test.tx).blockNumber,
    toBlock: web3.eth.getTransactionReceipt (test.tx).blockNumber
  }).get ();
assert (
  'resultEvents.length == 1',
  resultEvents.length == 1);
assert (
  '!resultEvents [0].args._value',
  !resultEvents [0].args._value);
assert (
  'test.standardTokenWrapper.allowance (test.dave.address, test.carol.address) == 1000',
  test.standardTokenWrapper.allowance (test.dave.address, test.carol.address) == 1000);
assert (
  'test.standardTokenWrapper.balanceOf (test.dave.address) == 0',
  test.standardTokenWrapper.balanceOf (test.dave.address) == 0);
}},
```

Full compliance to ERC-20 and predictable behavior

```
function transferFrom (address _from, address _to, uint256 _value)
returns (bool success) {
    if (allowances [_from][msg.sender] < _value) return false;
    if (accounts [_from] < _value) return false;
    allowances [_from][msg.sender] =
        safeSub (allowances [_from][msg.sender], _value);
    if (_value > 0 && _from != _to) {
        accounts [_from] = safeSub (accounts [_from], _value);
        accounts [_to] = safeAdd (accounts [_to], _value);
        Transfer (_from, _to, _value);
    }
    return true;
}
```



Deployment and Administration:
Dedicated UI



UI for smart contract interaction

Smooth user experience is crucial for security


- Runs in browser, Mist is not necessary;
- Connects to local or remote node;
- Accounts and/or passwords synced across devices;
- Plugin support to create administrator consoles.

 **Accounts**

 **Tokens**

 **Wallets**

 **Contracts**

 **Deploy contract**

 **Sign message**

 **Settings**

ACCOUNTS



Primary Test Account ⋮

0xe6a8f196629574cd13a91052...

Balance

9,892.80

[VIEW DETAILS](#)



Test wallet ⋮

0xb32933bd5c920d5fd08320c2...

Balance

129.92

[VIEW DETAILS](#)



Secondary Test Account ⋮

0xc23aac9593ae6bf1ba600073...

Balance

0.00








[VIEW DETAILS](#)





Example: Smart Token Wallet

- Connects to a remote node (Amazon EC2);
- Static file, works in any browser/OS locally.
- For users:
 - Balance and exchange rates for tokens alone and combined;
 - Approve and allowance functionality;
- For token administrators:
 - Fool-proof token resupply;
 - Freezing transfers;
 - Multisig administration.

-  Accounts
-  Tokens
-  Wallets
-  Contracts
-  Deploy contract
-  Sign message
-  Settings

COINS

Name	Balance	
Ether	9,892.80	TRANSFER

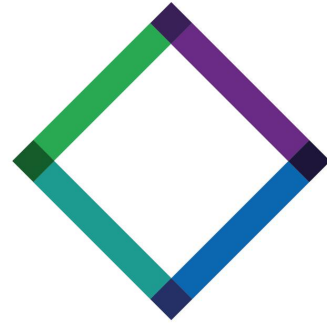
TOKENS

[MANAGE TOKENS](#)

Token	Balance	Allowance	Approved	Total		
Electoral token	528.00	0.00	30.00	498.00	TRANSFER	APPROVE
Square meter token	1,000.00	0.00	0.00	1,000.00	TRANSFER	APPROVE
UN voting token	10,000.00	0.00	0.00	10,000.00	TRANSFER	APPROVE
Moon Square Mile token	10,000.00	0.00	0.00	10,000.00	TRANSFER	APPROVE



Secure code, smart UI?



ABDK

www.abdk.consulting