Università degli Studi di Salerno
Dipartimento di Informatica
Corso di Laurea Magistrale in Informatica

CORSO DI
SICUREZZA DEI DATI

Anno Accademico 2024/2025
Proff. A. De Santis e C. Esposito

Simulazione Appello
Durata della prova: 60 minuti

Testo della prova

ESERCIZIO 1 [6 punti]
Descrivere il significato del seguente script BitCoin:

2DUP EQUAL NOT VERIFY SHA256 SWAP SHA256 EQUAL

Alternativa richiesta a lezione:

DUP HASH160 62e907b15cbf27d5425399ebf6f0fb50ebb88f18
EQUALVERIFY CHECKSIG

2 3 ADD 6 EQUAL

ESERCIZIO 2 [6 punti]
Descrivere cosa realizza il seguente smart contract in Solidity:

```
pragma solidity ^0.4.21;

contract Coin {
    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    function Coin() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        if (msg.sender != minter) return;
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        if (balances[msg.sender] < amount) return;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

Alternativa richiesta a lezione:

```
pragma solidity ^0.4.22;

contract Ballot {
    struct Voter {
        uint weight;
        bool voted;
        address delegate;
        uint vote;
    }

    struct Proposal {
        bytes32 name;
        uint voteCount;
    }

    address public chairperson;

    mapping(address => Voter) public voters;

    Proposal[] public proposals;

    constructor(bytes32[] proposalNames) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;

        for (uint i = 0; i < proposalNames.length; i++) {
            proposals.push(Proposal({
                name: proposalNames[i],
                voteCount: 0
            }));
        }
    }

    function giveRightToVote(address voter) public {
        require(
            msg.sender == chairperson,
            "Only chairperson can give right to vote."
        );
        require(
            !voters[voter].voted,
            "The voter already voted."
        );
        require(voters[voter].weight == 0);
        voters[voter].weight = 1;
    }

    function delegate(address to) public {
        Voter storage sender = voters[msg.sender];
        require(!sender.voted, "You already voted.");

        require(to != msg.sender, "Self-delegation is disallowed.");

        while (voters[to].delegate != address(0)) {
            to = voters[to].delegate;

            require(to != msg.sender, "Found loop in delegation.");
        }

        sender.voted = true;
        sender.delegate = to;
        Voter storage delegate_ = voters[to];
        if (delegate_.voted) {
            proposals[delegate_.vote].voteCount += sender.weight;
        } else {
            delegate_.weight += sender.weight;
        }
    }

    function vote(uint proposal) public {
        Voter storage sender = voters[msg.sender];
        require(!sender.voted, "Already voted.");
        sender.voted = true;
        sender.vote = proposal;

        proposals[proposal].voteCount += sender.weight;
    }

    function winningProposal() public view
            returns (uint winningProposal_)
    {
        uint winningVoteCount = 0;
        for (uint p = 0; p < proposals.length; p++) {
            if (proposals[p].voteCount > winningVoteCount) {
                winningVoteCount = proposals[p].voteCount;
                winningProposal_ = p;
            }
        }
    }

    function winnerName() public view
            returns (bytes32 winnerName_)
    {
        winnerName_ = proposals[winningProposal()].name;
    }
}
```

Alternativa:
Descrivere quale design pattern è implementato in questo smart contract in Solidity e quali sono le caratteristiche:

```
contract Donation_distributor {
    function donate(address addr) payable public {
       require(addr != address(0));
       require(msg.value != 0);
       uint balanceBeforeTransfer = this.balance;
       uint transferAmount;

       if (addr.balance == 0) {
           transferAmount = msg.value;
       } else if (addr.balance < msg.sender.balance) {
           transferAmount = msg.value / 2;
       } else {
           revert();
       }

       addr.transfer(transferAmount);
       assert(this.balance == balanceBeforeTransfer - transferAmount);
    }
}
```

ESERCIZIO 3 [6 punti]
Descrivere cosa realizza il seguente chaincode in Go:

```
package main
import (
   "fmt"
   "github.com/hyperledger/fabric/core/chaincode/shim"
   "github.com/hyperledger/fabric/protos/peer"
)

type SampleChaincode struct {

}

func (t *SampleChaincode) Init(stub shim.ChaincodeStubInterface)
peer.Response {
   args := stub.GetStringArgs()
   if len(args) != 2 {
      return shim.Error("Incorrect arguments. Expecting a key and a
value")
   }
   err := stub.PutState(args[0], []byte(args[1]))
   if err != nil {
      return shim.Error(fmt.Sprintf("Failed to create asset: %s",
args[0]))
   }
   return shim.Success(nil)
}
func (t *SampleChaincode) Invoke(stub shim.ChaincodeStubInterface)
peer.Response {
   fn, args := stub.GetFunctionAndParameters()
   var result string
   var err error
   if fn == "set" {
      result, err = set(stub, args)
   } else {
      result, err = get(stub, args)
   }
   if err != nil {
      return shim.Error(err.Error())
   }
   return shim.Success([]byte(result))
}
func set(stub shim.ChaincodeStubInterface, args []string) (string,
error) {
   if len(args) != 2 {
      return "", fmt.Errorf("Incorrect arguments. Expecting a key and
a value")
   }
   err := stub.PutState(args[0], []byte(args[1]))
   if err != nil {
      return "", fmt.Errorf("Failed to set asset: %s", args[0])
   }
   return args[1], nil
}
func get(stub shim.ChaincodeStubInterface, args []string) (string,
error) {
   if len(args) != 1 {
      return "", fmt.Errorf("Incorrect arguments. Expecting a
key")
   }
   value, err := stub.GetState(args[0])
   if err != nil {
      return "", fmt.Errorf("Failed to get asset: %s with error: %s",
args[0], err)
   }
   if value == nil {
      return "", fmt.Errorf("Asset not found: %s", args[0])
   }
   return string(value), nil
}

func main() {
   err := shim.Start(new(SampleChaincode))
   if err != nil {
      fmt.Println("Could not start SampleChaincode")
   } else {
   fmt.Println("SampleChaincode successfully  started")
   }
}
```

ESERCIZIO 4 [6 punti]

Quale vulnerabilità è presente in questo smart contract in Solidity? Descrivere una possibile soluzione:

```
contract EtherStore {
    uint256 public withdrawalLimit = 1 ether;
    mapping(address => uint256) public lastWithdrawTime;
    mapping(address => uint256) public balances;

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

    function withdrawFunds (uint256 _weiToWithdraw) public {
        require(balances[msg.sender] >= _weiToWithdraw);
        // limit the withdrawal
        require(_weiToWithdraw <= withdrawalLimit);
        // limit the time allowed to withdraw
        require(now >= lastWithdrawTime[msg.sender] + 1 weeks);
        require(msg.sender.call.value(_weiToWithdraw)());
        balances[msg.sender] -= _weiToWithdraw;
        lastWithdrawTime[msg.sender] = now;
    }
```

```
 }
```

ESERCIZIO 5 [6 punti]

Descrivere il seguente chaincode go:

```
package main

import (
    "fmt"
    "time"
)

func worker(done chan bool) {
    fmt.Print("working...")
    time.Sleep(time.Second)
    fmt.Println("done")

    done <- true
}

func main() {
    done := make(chan bool, 1)
    go worker(done)
    <-done
}
```

Alternativa richiesta a lezione:

```
package main


import "fmt"

func ping(pings chan<- string, msg string) {
    pings <- msg
}

func pong(pings <-chan string, pongs chan<- string) {
    msg := <-pings
    pongs <- msg
}

func main() {
    pings := make(chan string, 1)
    pongs := make(chan string, 1)
    ping(pings, "passed message")
    pong(pings, pongs)
    fmt.Println(<-pongs)
}
```

Alternativa richiesta a lezione:

```
package main

import (
    "fmt"
    "time"
)
```

```
func main() {

    c1 := make(chan string)
    c2 := make(chan string)

    go func() {
        time.Sleep(1 * time.Second)
        c1 <- "one"
    }()
    go func() {
        time.Sleep(2 * time.Second)
        c2 <- "two"
    }()


     fmt.Println("received", <-c2)
     fmt.Println("received", <-c1)
}
```