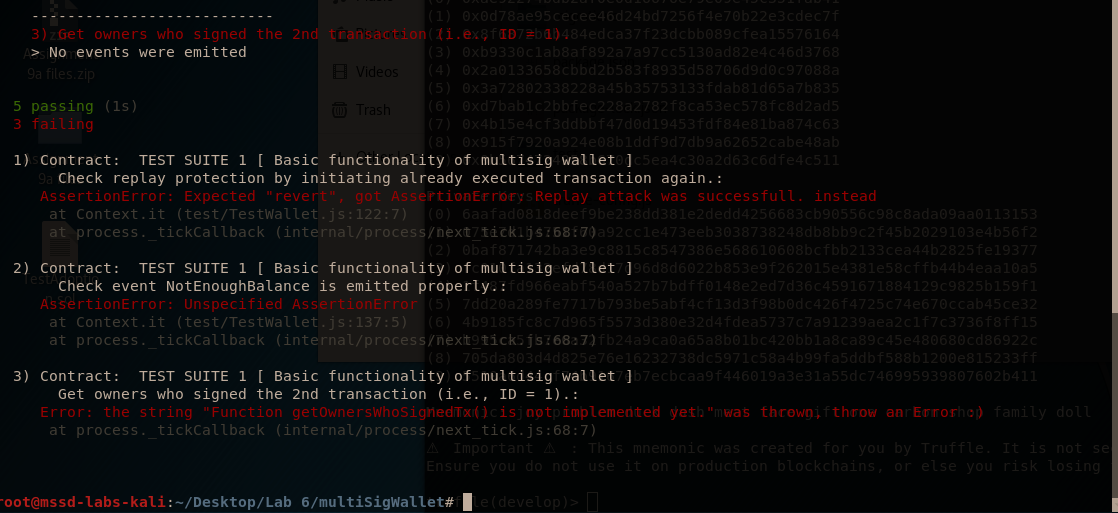
**At all times you are only required to modify only these 2 files.**

1) smart contract in MultisigWallet.sol and

2) wallet\_config.js that adjust parameters of the wallet,

**5. Run blockchain and tests**

****

**6. Task 1 - Protect against Replay Attack**

modifier txnNotExecuted(uint transactionId) {

if (transactions[transactionId].executed)

revert("Transaction already executed.");

\_;

}

We create the modifier as above and call it under function executeTransaction, as below:

function executeTransaction(uint transactionId)

public

// check for replay

**txnNotExecuted(transactionId)**

**7. Task 2 - Transform the 2-of-2 Wallet to n-of-m Multisig**

We transform the constructor:

constructor(address[] memory \_owners, uint \_requiredSigs) public

checkValidSettings(\_owners.length, \_requiredSigs)

{

// TASK 2: Modify this constructor to fit n-of-m scheme, i.e., an

for(uint i = 0; i < \_owners.length; i++){

checkNotNull(\_owners[i]);

for (uint j = i+1; j < \_owners.length; j++){

if(\_owners[i] == \_owners[j]){

revert("A repeated owner passed.");

}

}

}

// save owners (m) and the minimum number of signatures (n)

for(uint i=0; i<\_owners.length; i++){

isOwner[\_owners[i]] = true;

}

owners = \_owners;

minSignatures = \_requiredSigs;

}

**Task 3 - Check Enough Balance of Contract Wallet**

// TASK 3: check whether the contract has enough balance and if not

// emit a new event called

// NotEnoughBalance(curBalance, requestedBalance)

if (isTxConfirmed(transactionId)) {

Transaction storage txn = transactions[transactionId];

uint curBalance = address(this).balance;

uint requestedBalance = txn.value;

if (requestedBalance > curBalance) {

emit NotEnoughBalance(curBalance, requestedBalance);

}

**Task 4 - Write a Function for Retrieval of Owners that Sign a Transaction**

function getOwnersWhoSignedTx(uint transactionId)

public view returns (address[] memory)

{

address[] memory signedOwners = new address[](

getSignatureCount(transactionId)

);

uint cursor = 0;

for (uint i = 0; i < owners.length; i++) {

if (signatures[transactionId][owners[i]]){

signedOwners[cursor] = owners[i];

cursor += 1;

}

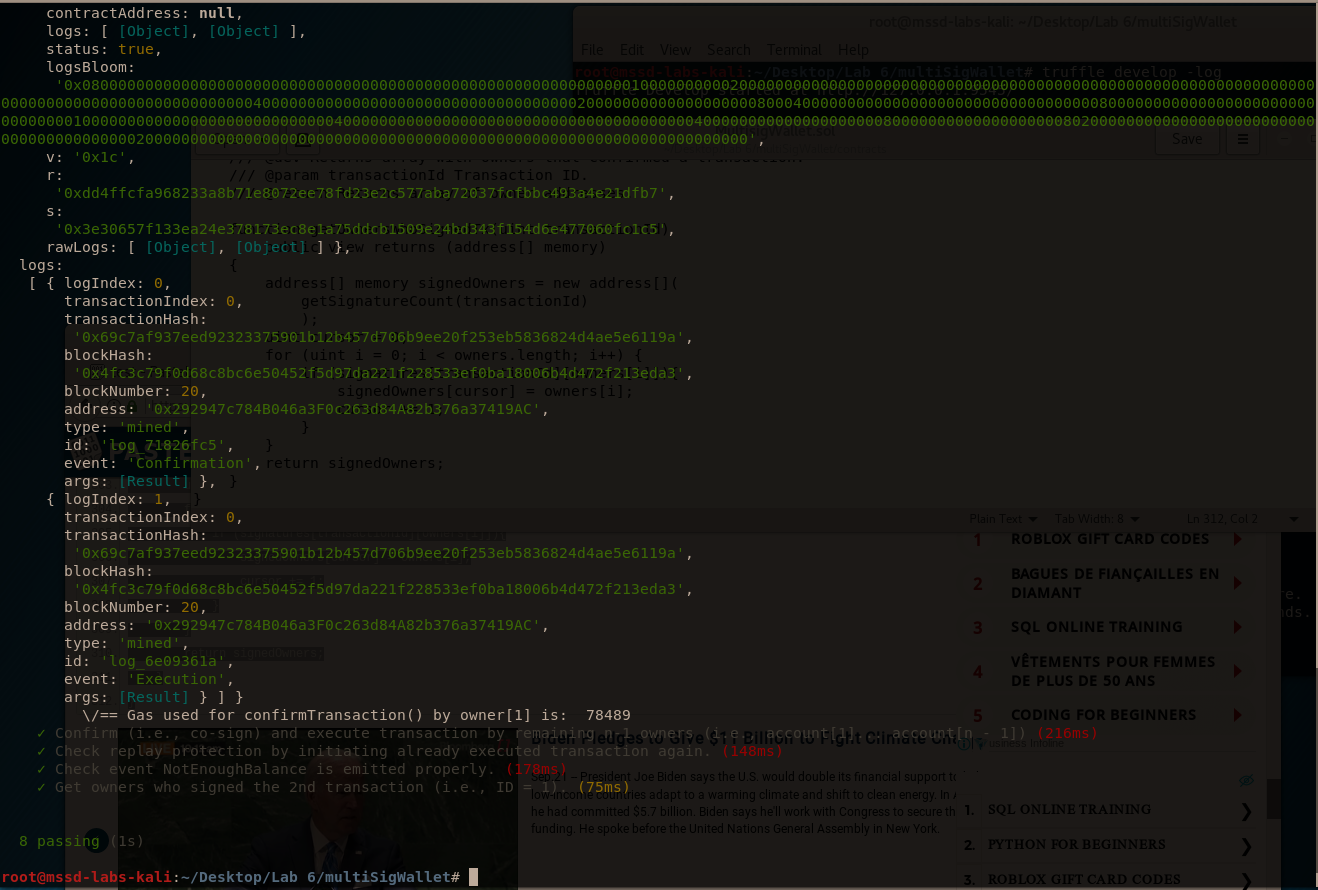
}

return signedOwners;

}

}

**All the tests clear:**

****

**Task 5 - Investigate the Receipt of a Transaction**

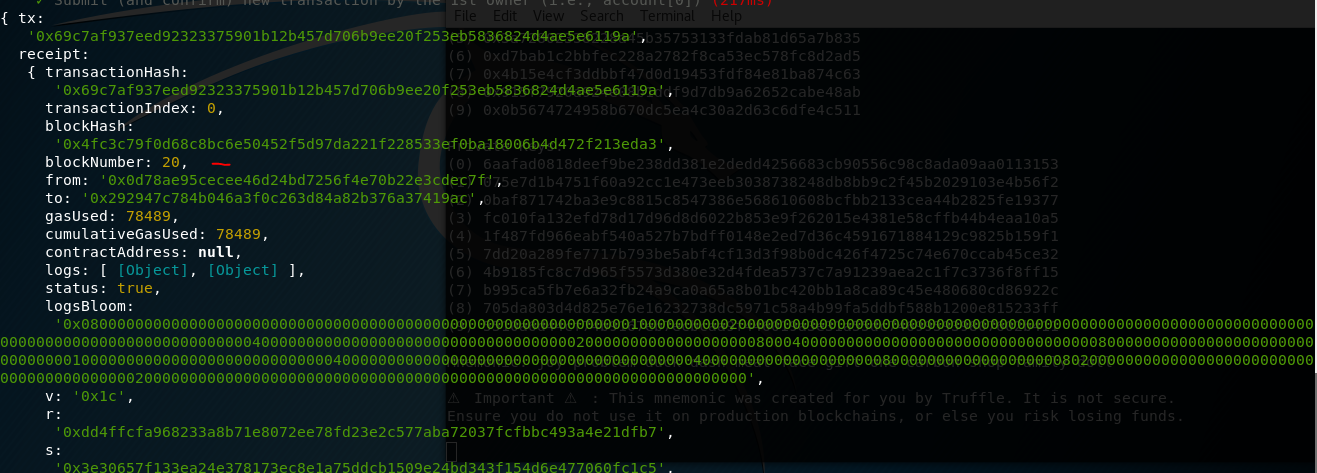
**Observe in the first terminal how much gas you pay for the deployment of your contract (ignore a deployment of migration contract that costs around 300K of gas).**

**What is the block number and how much would you pay for it in USD?**

**Some of the tests print the gas consumption of a few function calls of smart contracts.**

**Try to print the full receipt of confirmation and submission calls and comment on the meaning of the most important fields seen there.**

**What events do you see there?**

****

* The contract was created in block number 20.
* 78489 gas was used.