NAMA : ARASY BAZWIR

NIM : 1103183236

Technical Documentation LAB 2 – Shared Wallet

Pada lab kedua ini akan mempelajari bagaimana cara sebuah Wallet menyimpan dana dan memungkinkan penggunanya untuk menarik kembali dananya, pengguna juga dapat memberikan izin kepada pengguna tertentu lainnya.

```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.1;

contract SharedWallet{

    function withdrawMoney(address payable _to, uint _amount) public{
        _to.transfer(_amount);
    }

    receive() external payable{

    }
}
```

Code berikut adalah sebuah *Smart Contract* dasar yang bisa menerima dan menarik sebuah Ether, tetapi sebenarnya masih belum bisa digunakan untuk kasus asli.

```
//SPDX-License-Identifier: MIT
pragma solidity 0.8.1;
contract SharedWallet {

    address owner;

    constructor() {
        owner = msg.sender;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "You are not allowed");
        _;
    }

    function withdrawMoney(address payable _to, uint _amount) public onlyOwner {
        _to.transfer(_amount);
    }

    receive() external payable {

    }
}
```

Code berikut menggunakan sebuah fungsi yang bisa membatasi penarikan dana kedalam *Wallet* pengguna.

```
//SPDX-License-Identifier: MIT

pragma solidity 0.8.1;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Own

contract SharedWallet is Ownable {
    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }

    function withdrawMoney(address payable _to, uint _amount) public onlyOwner {
        _to.transfer(_amount);
    }

    receive() external payable {
    }
}
```

Code berikut memecah beberapa bagian dan menggunakan *Smart Contract* yang telah di audit dari *Open Zeppelin*.

```
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Own

contract SharedWallet is Ownable {

    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }

    mapping(address => uint) public allowance;

    function addAllowance(address _who, uint _amount) public onlyOwner {
        allowance[_who] = _amount;
    }

    modifier ownerOrAllowed(uint _amount) {
        require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
        _;
    }

    receive() external payable {
    }

    function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
        allowance[_who] -= _amount;
    }

    function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
        require(_amount <= address(this).balance, "Contract doesn't own enough money");
        if(!isOwner()) {
            reduceAllowance(msg.sender, _amount);
        }
        _to.transfer(_amount);
    }
```

Code berikut menambahkan pemetaan sehingga dapat menyimpan alamat jumlah unsigned int. ini akan menjadi seperti array yang menyimpan sebuah alamat ke nomor tertentu. Fungsi baru juga di tambahkan sehingga seseorang bisa tahu berapa banyak penarikan yang bisa di tarik.