



# Lesson 16

---

**UPGRADING HARDHAT**

---

Hikmah Nisya - 1103184094  
Radzis Araaf Jaya Jamaludin - 1103184234  
Raudhatul Rafiqah Assyahiddini - 1103180225

```
C:\WINDOWS\system32>npm install --save-dev hardhat  
added 270 packages, and audited 367 packages in 2m  
69 packages are looking for funding  
  run `npm fund` for details  
found 0 vulnerabilities
```

Sebelum mengupgrade Hardhat, kita asumsikan bahwa user sudah menginstall Hardhat pada CMD, di Lesson ini saya menginstall dengan npm

---



Dengan mengupgrade smart contract kita, kita dapat mengupdate parameter yang sebelumnya tidak bisa diimplementasikan seperti tidak bisa menambah sebuah logika pada contract kita ataupun “adding storage”



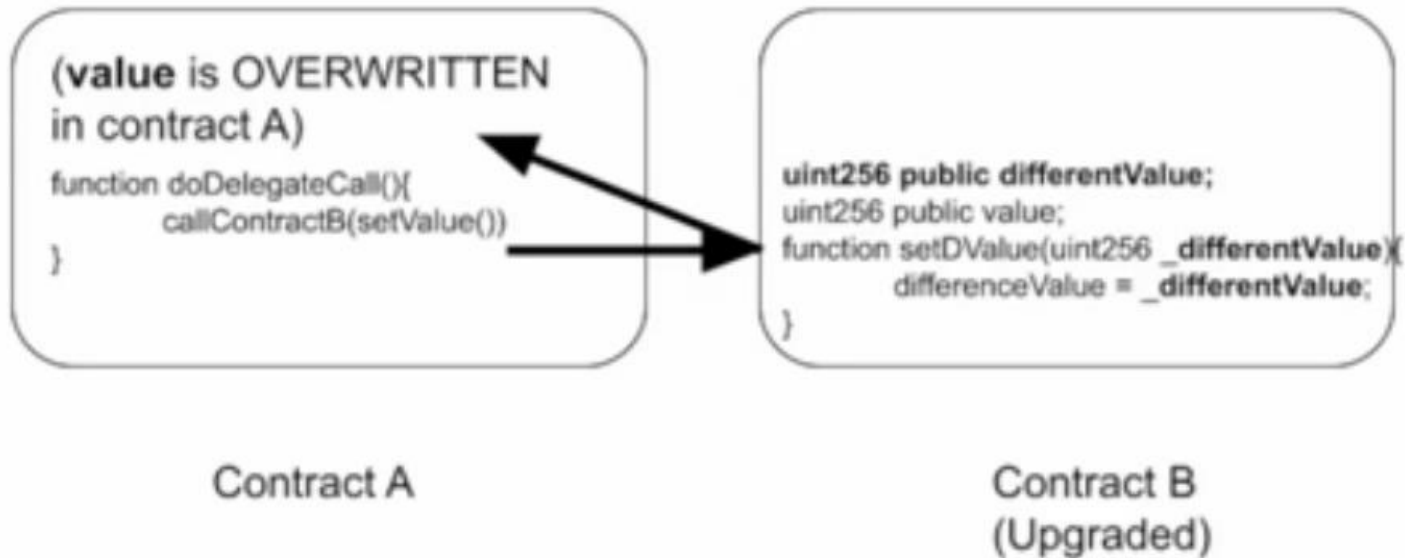
Untuk mengupgrade smart contract merupakan hal yang simple, akan tetapi tidak flexible



Apakah mengupgrade smart contract menghilangkan esensi immutability? Tidak, karena dengan mengupgrade smart contract, berarti semakin berkurang vulnerabilitas keamanan yang ada pada smart contract kita



Kekurangan dari dengan mengupgrade smart contract adalah meyakinkan user untuk pindah ke smart contract baru, dan pastinya address baru untuk semua node yang terkoneksi kedalam jaringan blockchain



Dan apakah migrasi smart contract menghilangkan value dari contract pertama kita?, Hal ini bisa di atasi dengan sebuah codingan yaitu delegatecall, yang mengubah/mengupgrade function pada smart contract pertama kita tanpa mengubah value dari msg.sender dan msg.value

---

```
contract A {
    uint public num;
    address public sender;
    uint public value;

    function setVars(address _contract, uint _num) public payable {
        // A's storage is set, B is not modified.
        (bool success, bytes memory data) = _contract.delegatecall(
            abi.encodeWithSignature("setVars(uint256)", _num)
        );
    }
}
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

// NOTE: Deploy this contract first
contract B {
    // NOTE: storage layout must be the same as contract A
    uint public num;
    address public sender;
    uint public value;



    function setVars(uint _num) public payable {
        num = _num;
        sender = msg.sender;
        value = msg.value;
    }
}
```

Untuk pengujian delegatecall, akan menggunakan 2 smart contract

---

# Jalankan contract B dahulu lalu Contract A

Deployed Contracts

▼ B AT 0XD91...39138 (MEMORY)  

**setVars** 455 ▼

**num**

o: uint256: 455

**sender**


o: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

**value**


o: uint256: 0



**setVars** ^

 **\_contract:** "0xd9145CCE52D386f25491"

**\_num:** 222

 **transact**

**num**

o: uint256: 222

**sender**

o: address: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db

**value**

o: uint256: 0

Keunikan dari delegate call adalah, walaupun nama variable pada contract berbeda dengan yang ada pada function ini, function tersebut akan menyesuaikan dengan variable yang ada pada contract

```
function setVars(uint _num) public payable {  
    num = _num;  
    sender = msg.sender;  
    value = msg.value;  
}  
}
```

---

Fungsi dari delegatecall pada codingan adalah seperti kita meng copy-paste function dari contract B ke pada Contract A, Function dibawah hanya tertulis pada contract B, tapi saat kita menggunakan delegate call pada Contract A, function ini tidak perlu diisi ulang