MMSeminar
Ethereum Smart Contract

7/21

Chihiro Kado, Chika Komiya, Koki Tejima, Yusho Yamaguchi

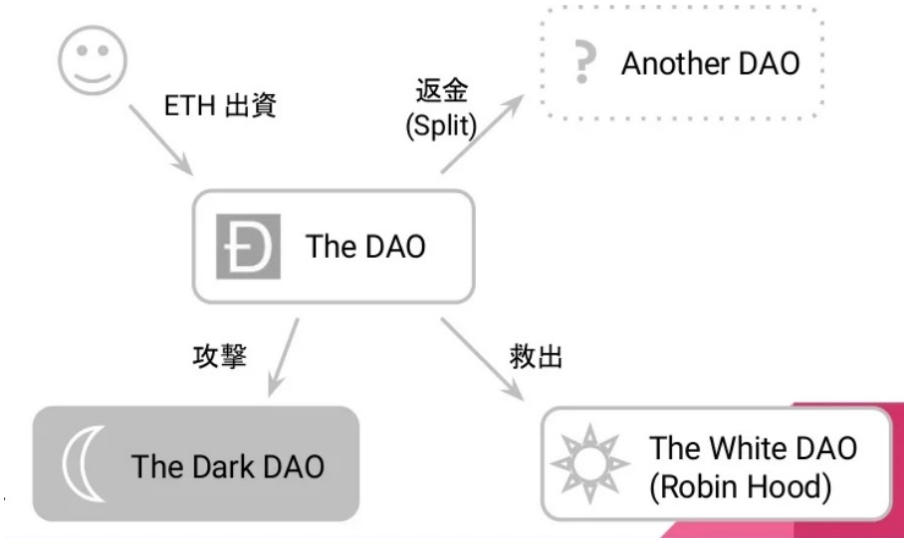
加道ちひろ　小宮千佳　手島宏貴　山口雄翔

Outline

- Overview
- Vulnerabilities
- Usage : NFT

Why Ethereum Smart Contract

- Popular Cryptocurrency
- Some incidents had happend



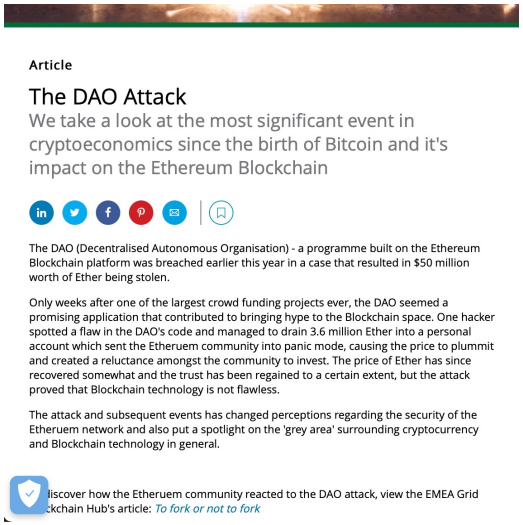
The Parity Wallet Hack Explained

JULY 19, 2017 | IN SECURITY AUDITS | BY SANTIAGO PALLADINO

> Thank you for your interest in this post! We're undergoing a rebranding process, so please excuse us if some names are out of date.

TL;DR

- A vulnerability was found on the Parity Multisig Wallet version 1.5+, that allowed an attacker to steal over 150,000 ETH (~30M USD).
- If you are using the affected wallet contract, make sure to move all funds to a different wallet immediately.
- The OpenZeppelin MultiSig wallet is unaffected by the vulnerability.

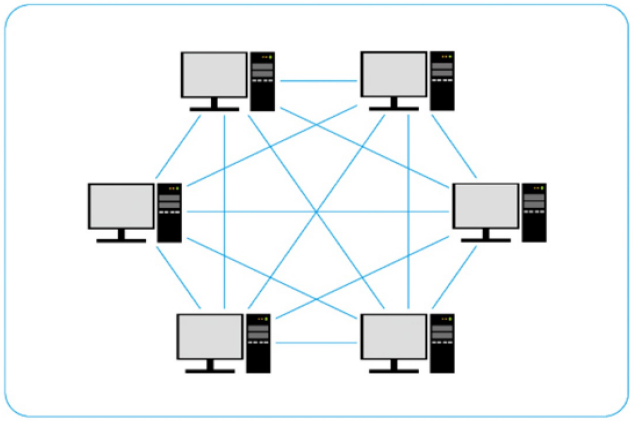


https://www.slideshare.net/YutoTakei/the-dao
https://blog.openzeppelin.com/on-the-parity-wallet-multisig-hack-405a8c12e8f7/
https://www2.deloitte.com/ie/en/pages/technology/articles/DAO-Attack-Analysis.html

Overview

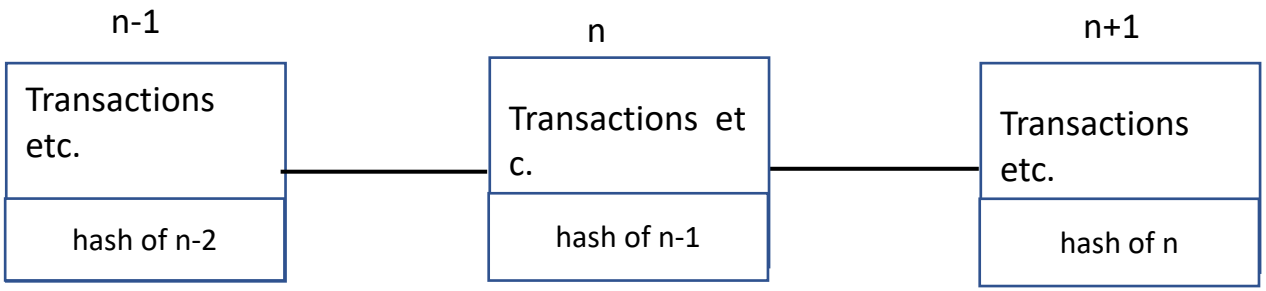
About BlockChain

- Mechanisms for storing data in a decentralized manner
- It is difficlut to manipulate the stored data
- The blockchain network uses a method called P2P (Peer to Peer)



- It is used in many kind of Cryptocurrency

What is BlockChain



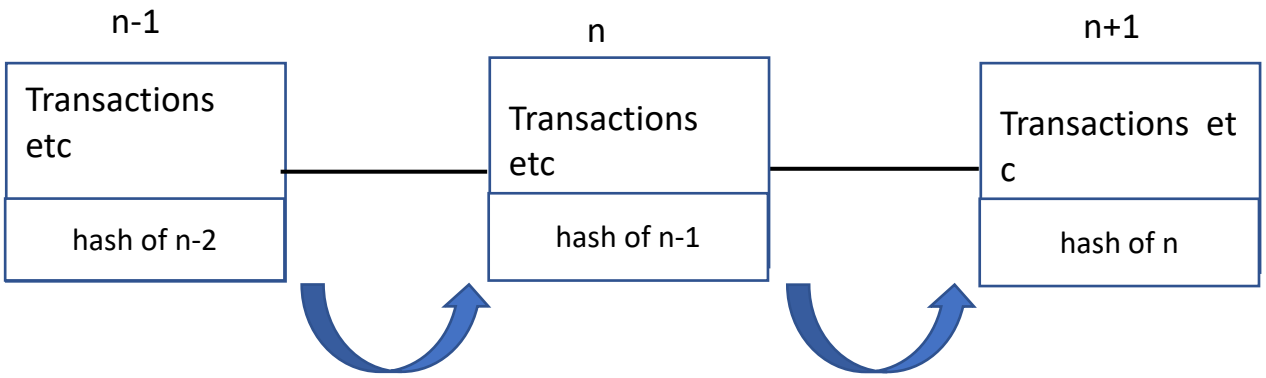
- Storing information in Blocks
- In Cryptocurrency, Transaction information etc.
- The same blockchain is shared by all participants

- Under certain conditions, a new block is generated

New block

- Mining is computation required to make a block which is admitted as appropriate
- One miner can create a block
- The miner receives a reward

Reliability of BlockChain



- Put a hash of the information in the previous block into the next block
- Rewriting is extremely difficult

- Digital Signature keep the validity

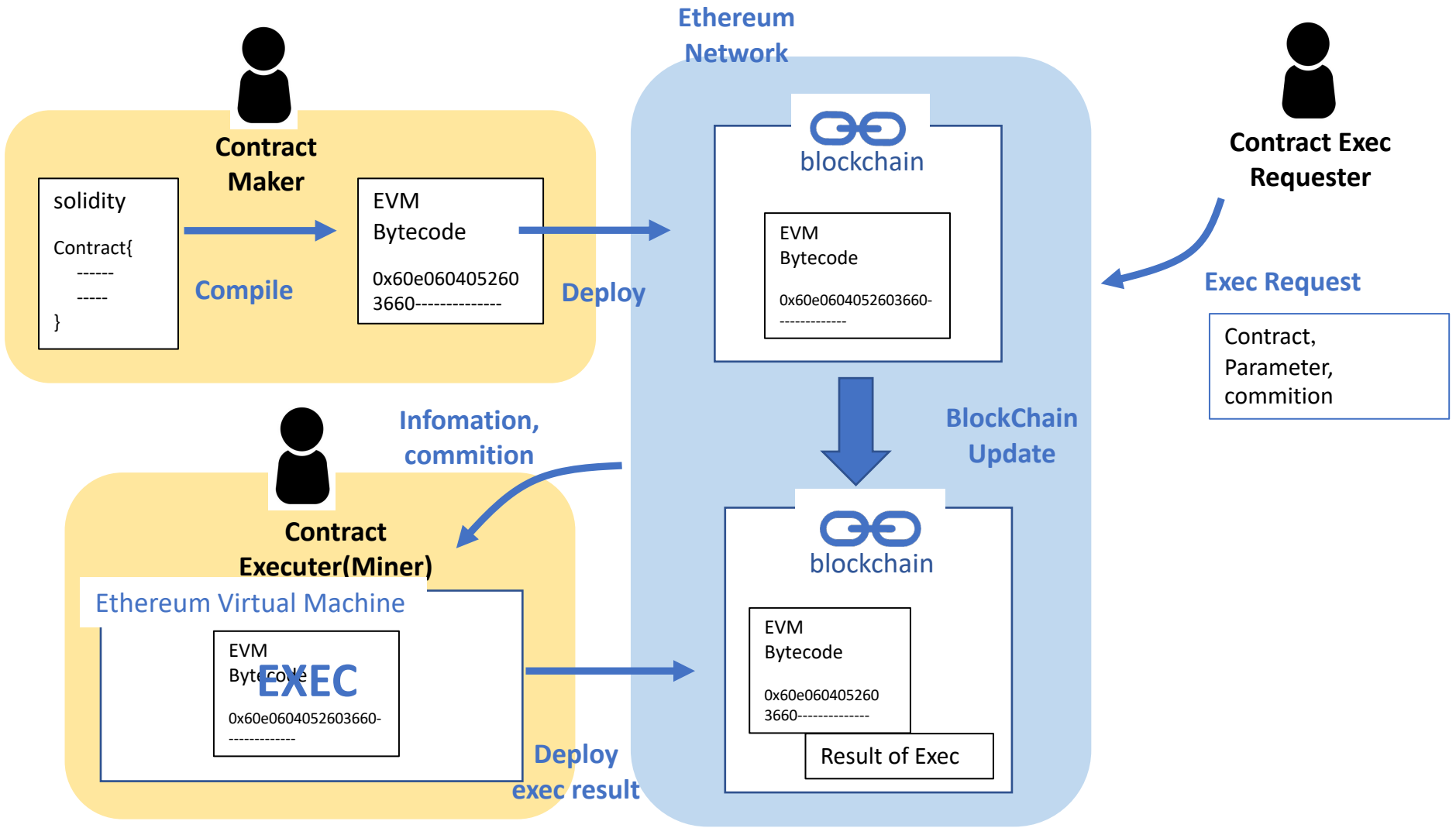
Ethereum Smart Contract

- A kind of Cryptocurrency
- Smart Contract is available in Ethereum
- What is Smart Contract
  - Programs which is incorporated in BlockChain
  - It is used in some Cryptocurrency like Ethereum
  - Modifying it is extremely difficult
  - It automatically execute contracts under specific conditions.

Terms of Ethereum

- Transaction
  - Information of Ethereum transaction which the miner put to block
- EVM
  - Virtual machines held by ethereum participants
  - EVM executes the contract's code
- ETH
  - Currency Units in Ethereum

Operating Principle



Example of Contract

```
pragma solidity ^0.4.21;

contract SampleToken {
    // 状態変数の宣言
    string public name;             // トークンの名前
    string public symbol;           // トークンの単位
    uint8 public decimals;          // 小数点以下の桁数
    uint256 public totalSupply;     // トークンの総量
    mapping (address => uint256) public balanceOf;      // 各アドレスの残高

    // イベント通知
    event Transfer(address indexed from, address indexed to, uint256 value);

    // コンストラクタ
    function SampleToken(uint256 _supply, string _name, string _symbol, uint8 _decimals) {
        balanceOf[msg.sender] = _supply;
        name = _name;
        symbol = _symbol;
        decimals = _decimals;
        totalSupply = _supply;
    }

    // 送金
    function transfer(address _to, uint256 _value) {
        // 送信アドレスと受信アドレスの残高を更新
        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;

        // イベント通知
        Transfer(msg.sender, _to, _value);
    }
}
```

https://www.tdi.co.jp/miso/blockchain-ethereum-solidity-remix

Gas

- We call virtual fuel used when executing contracts in ethereum 'Gas'

- By deciding upper limit of Gas which can be used in a smart contract, consumption of computing resources is limited

Vulnerabilities

- DELEGATECALL
- Default Visibilities
- Unchecked CALL Return Values
- Reentrancy
- Arithmetic Over/Underflows
- Denial of Service(DoS)

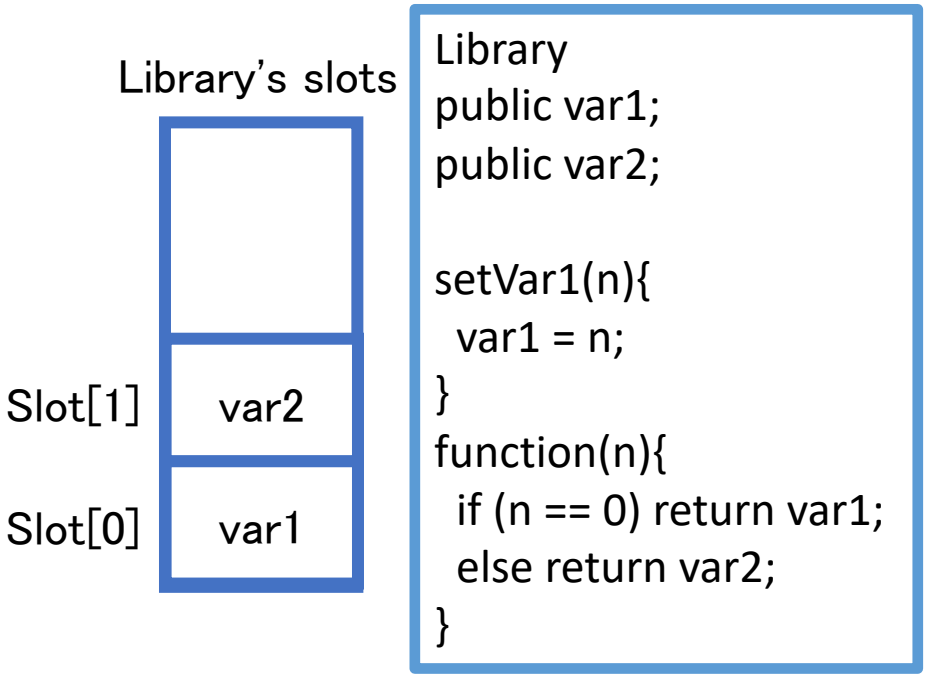
DELEGATECALL
What is DELEGATECALL

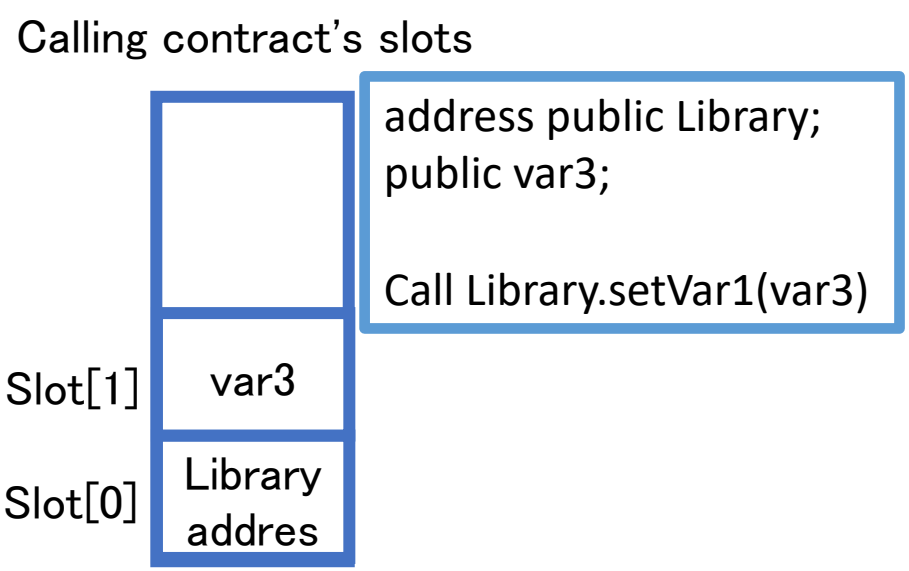
- **CALL** and **DELEGATECALL** opcodes are useful in allowing Ethereum developers to modularize their code.
- **DELEGATECALL** opcode's feature
  - the called code runs in the context of the calling contract
  - msg.sender and msg.value remain unchanged

- The feature enables the implementation of *libraries* !

DELEGATECALL
The Vulnerability

- When run in the context of another application, new vulnerabilities can arise
- State or storage variables are placed into *slots* sequentially as they are introduced in the contract





DELEGATECALL Real-World Example

- About $300Million worth of ETH is frozen and lost
- The WalletLibrary contract could be initialized and become owned
- A user became owned and called the kill function
- All of their functionality, including to withdraw ether was lost
- All ether in all Parity multisig wallets of this type instantly became lost or permanently unrecoverable

[DEC] https://medium.com/chain-cloud-company-blog/parity-multisig-hack-again-b46771eaa838

Default Visibilities
What is Visibilities

- The visibility determines whether a function can be called
- Functions in Solidity have visibility specifiers
- Four visibility specifiers
  - **public** : the contract interface and can be either called internally or via message calls
  - **external** : the contract interface, which means they can be called from other contracts and via transactions
  - **internal** : only be accessed from whithin the current contract or contracts deriving from it.
  - **private** : only be accessed from the current contract

restricted

Default Visibilities
The Vulnerability

- Default visibility of functions is **public**
- The issue arises when developers mistakenly omit visibility specifiers on functions

Preventative Techiniques

- Always specify the visibility of all functions
- Recent versions of compiler show a warning for functions that have no explicit visibility

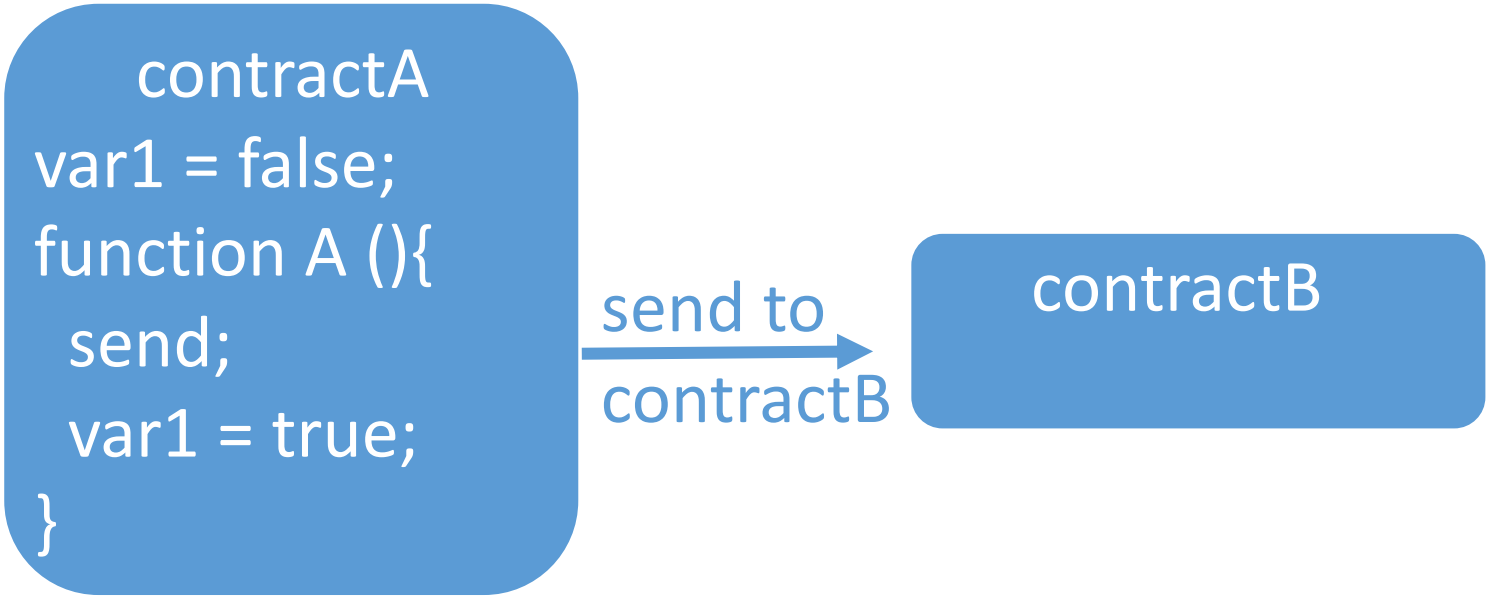
Default Visibilities Real-World Example

- About $31Million worth of ETH was stolen
- InitFunction was accidentally left **public**
- InitFunction sets the owners for the multisig wallet
- Attacker reseted the ownership to the attacker's address
- The attacker drained the wallets of all their ether

[DEV]https://hackingdistributed.com/2017/07/22/deep-dive-parity-bug/

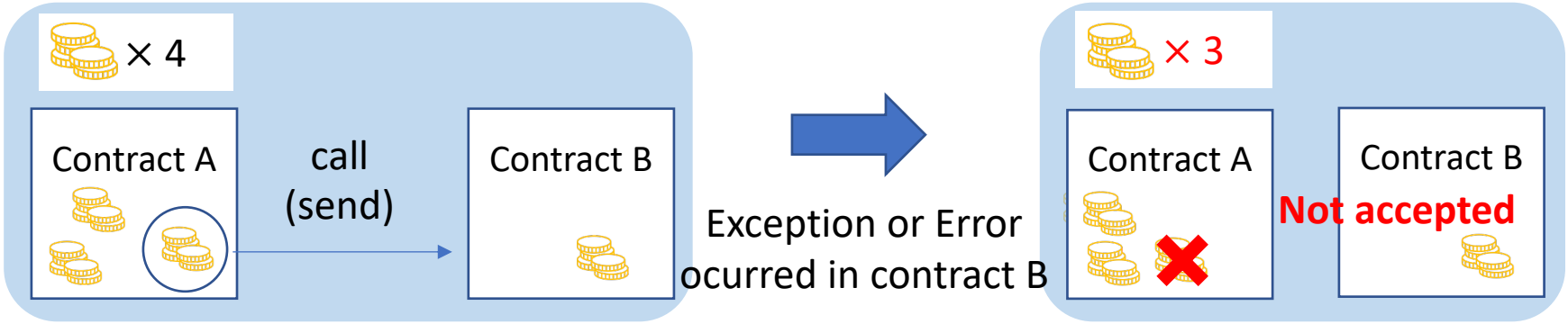
Unchecked CALL Return Values
What is Unchecked CALL Return Values

- The **call** and **send** functions return a Boolean
- Boolean indicates whether the call succeeded or failed
- If the call failed the function will simply return **false**, error will not occur



Unchecked CALL Return Values The Vulnerability

- Vulnerability



- Preventative Techniques
  - use transfer function
  - check the return values of send function

Unchecked CALL Return Values Real-World Example

EtherPot

- Etherpot was a smart contract lottery

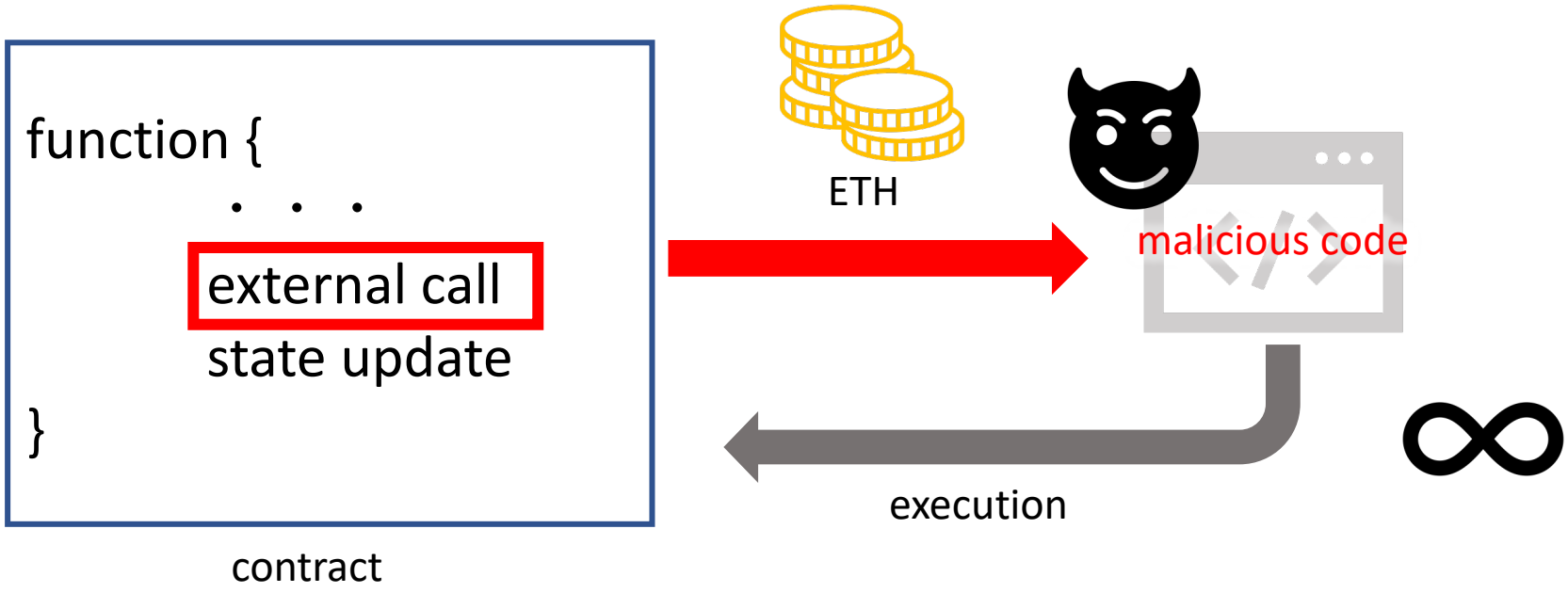
King of the Ether

- You can get the Throne of the King

[ETP] https://github.com/etherpot/contract/blob/master/app/contracts/lotto.sol
[KOE] https://www.kingoftheether.com/postmortem.html

Reentrancy
The Vulnerability

Occurred when a contract sends ether to an unknown address.



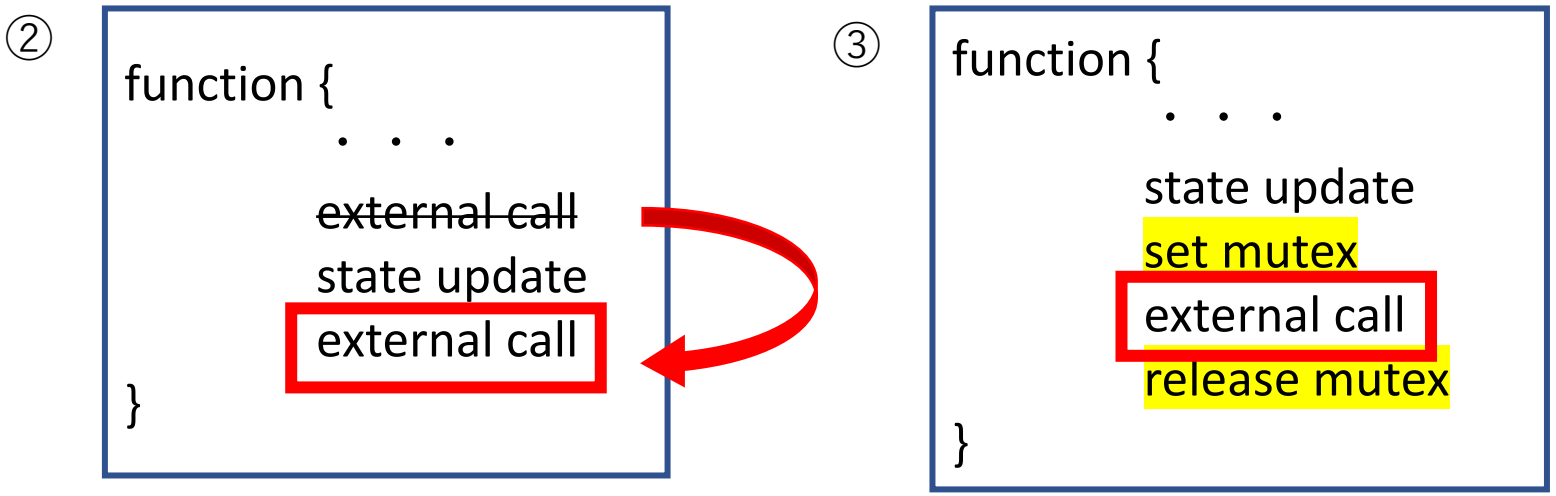
Performing operations not expected by the developer

Reentrancy
Preventative Techniques

① Use the built-in transfer function when sending ETH to external contracts.

② Ensure that all logic that changes state variables happens before ether is sent out of the contract.

③ Introduce a mutex

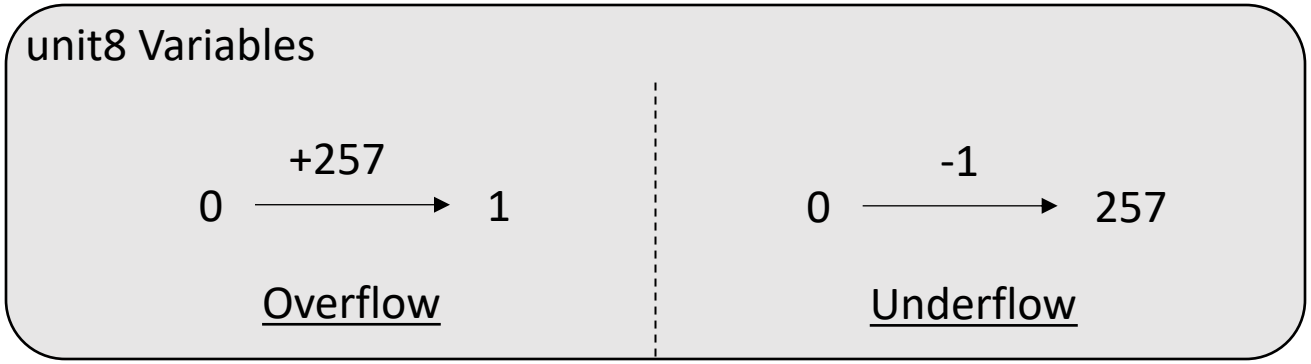


Reentrancy
Real-World Example : The DAO

- One of the major hacks that occurred in the early development of Ethereum.
- Approximately 3.6 million ETH(approximately 5.2 billion yen) was stolen.
- Reentrancy played a major role in the attack.
- Ultimately led to the hard fork that created Ethereum Classic(ETC).

Arithmetic Over/Underflows
The Vulnerability

- Occurred when an operation is performed that requires a fixed-size variable to store a number that is outside the range of the variable's data type.
- The Ethereum Virtual Machine specifies fixed-size data types for integers.
- Numerical gotchas allow attackers to misuse code and create unexpected logic flows.



Arithmetic Over/Underflows Preventative Techniques

- Use or build mathematical libraries that replace the standard math operators addition, subtraction, and multiplication.

  SafeMath library of OpenZeppelin

  Avoid under/overflow vulnerability

- Use Effective compiler

  Cancels execution and returns to the original state

Arithmetic Over/Underflows Real-World Example

PoWHC (Proof of Weak Hands Coin)

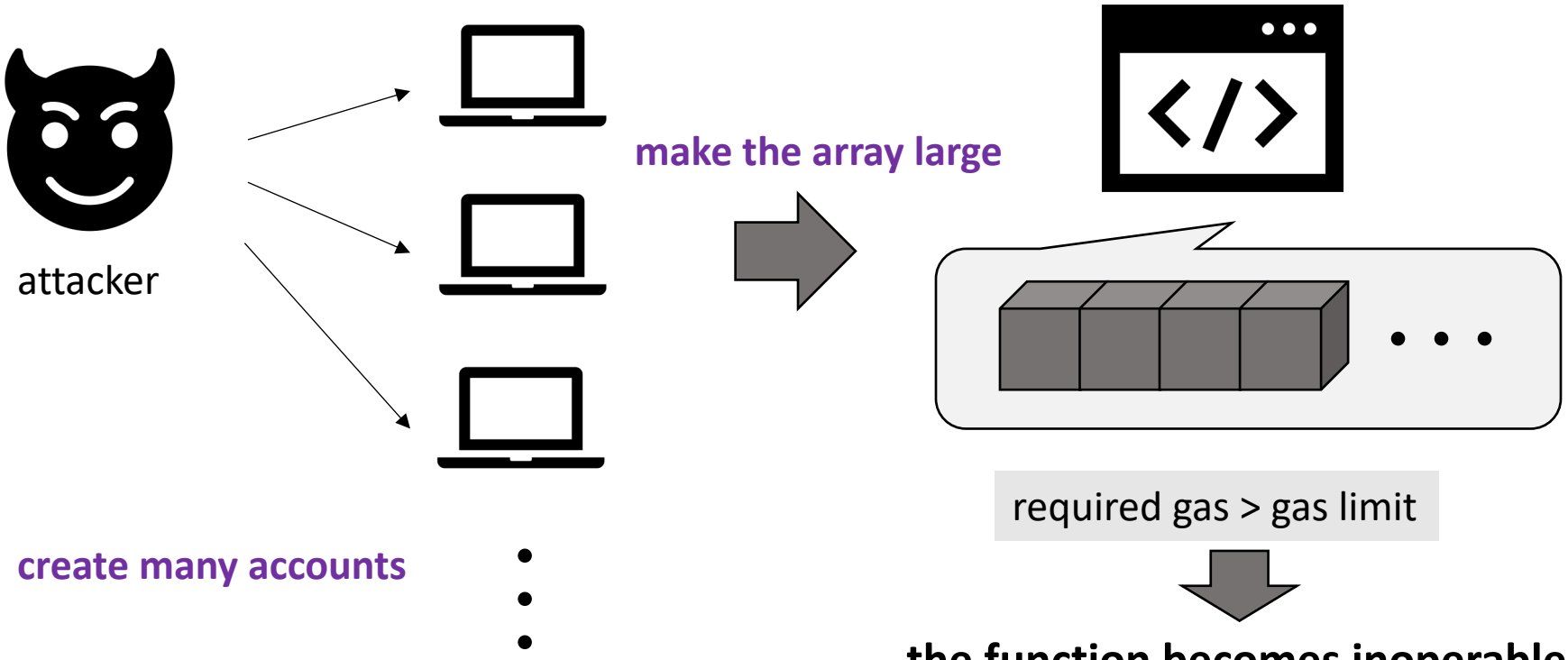
- 866 ETH were liberated from the contract.
- Cause: The author did not take action.

Batch Transfer Overflow

- The implementation of a batch Transfer() function into a group of ERC20 token contracts.
- The implementation contained an overflow vulnerability.

Denial of Service (DoS)
The Vulnerability

Attacks where users can reader a contract inoperable for a period of time, or in some cases permanently.



Denial of Service (DoS) Preventative Techniques

Contracts should not loop through data structures that can be artificially manipulated bay external users.

```
contract DistributeTokens {
        address[] investors;
         . . .
        function {
                  . . .
                for( i=0; i < investors.length; i++) {
                            . . .
                }
        }
}
```

→Process individually using the withdraw function

Denial of Service (DoS)
Real-World Example : GovernMental

- GovernMental was susceptible to the DoS vulnerabilities.
- The 1,100 ETH were finally obtained with a transaction that used 2.5M gas

Usage：

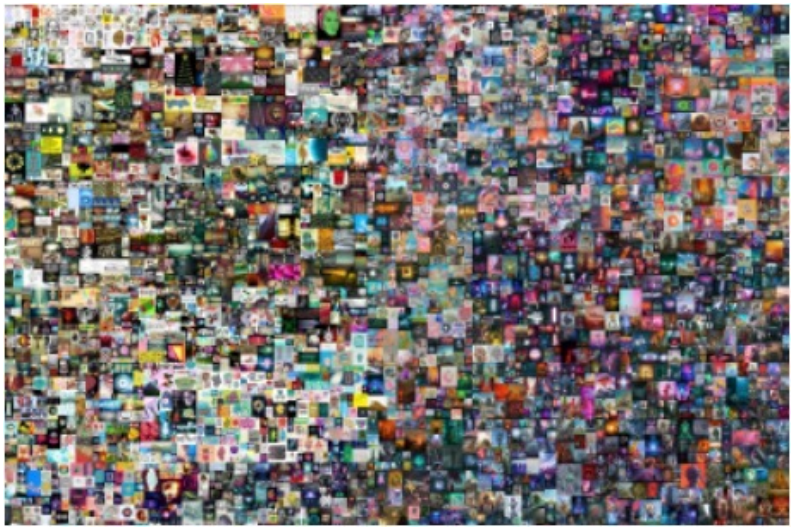
NFT（Non-Fungible Token）

What is NFT

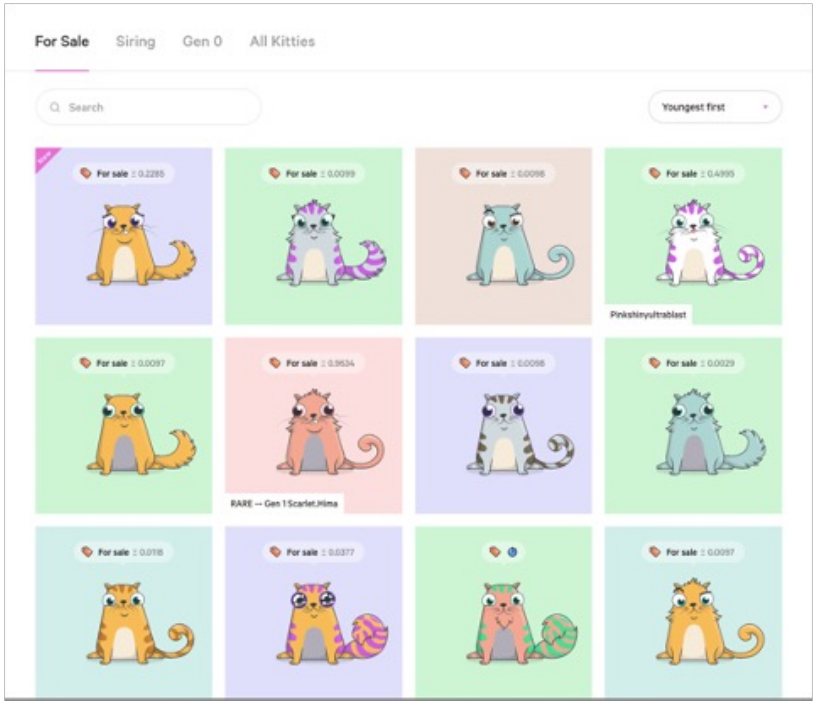
- Token
  - Fungible Token: ETH(cryptocurrency of Ethereum)
    - Indistinguishable
    - Equivalent
  - Non-Fungible Token: digital art, event chicket, game character data, etc.
    - Unique
    - Unexchangeable
- Feature of NFT
  - Link to digital data
  - Managed by smart contracts
  - Prove the existence and ownership

What is NFT

NFT Art



NFT Game



Token standards on Ethereum

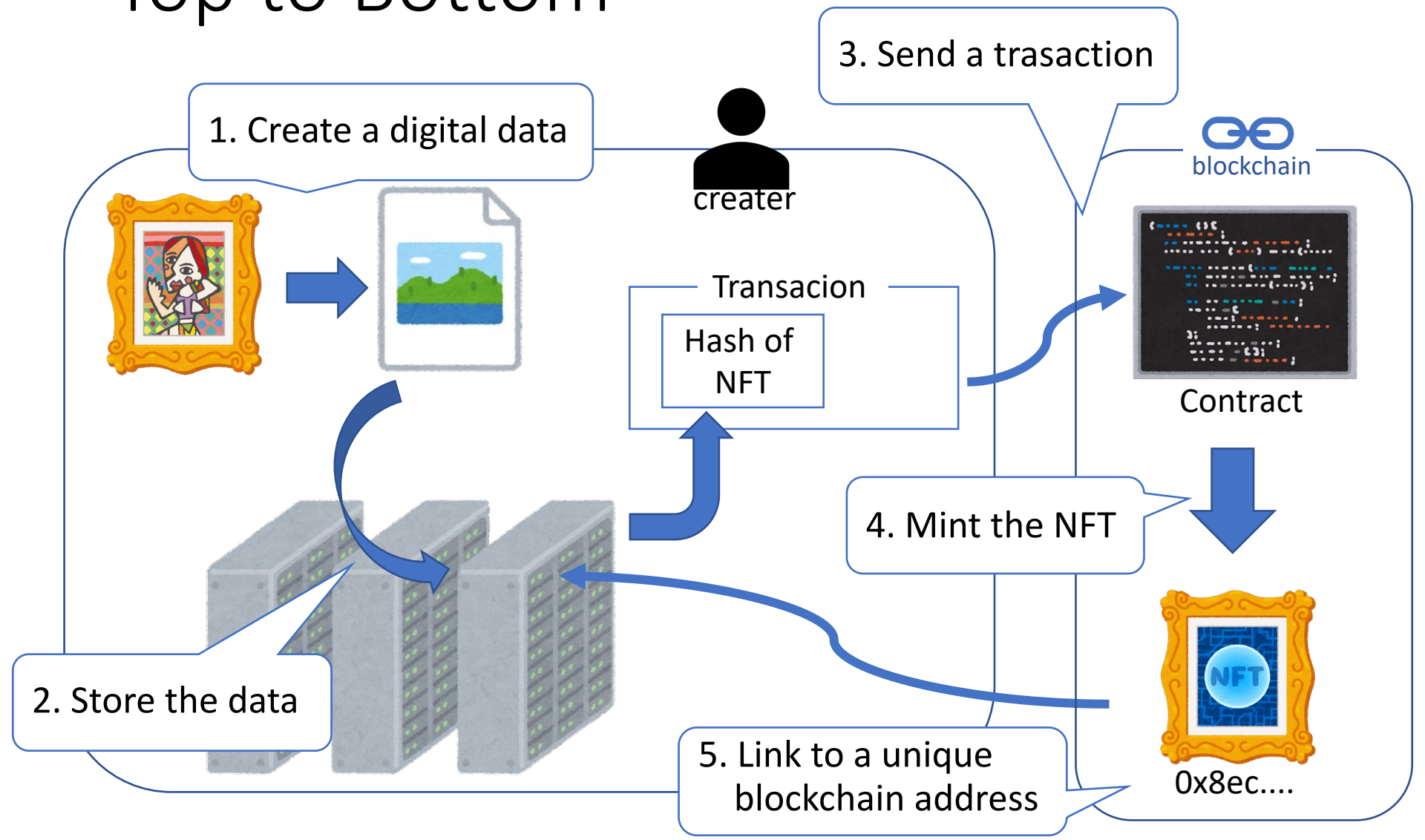
- ERC-20(2015)
  - Standard for fungible tokens
  - Function for representing amount of tokens
- ERC-721(2018)
  - Standard for non-fungible tokens
  - Function for managing ownerships
- ERC-1155(2019)
  - Standard for fungible and non-fungible tokens
  - One contract can mint and handle both types of token

How to create NFTs

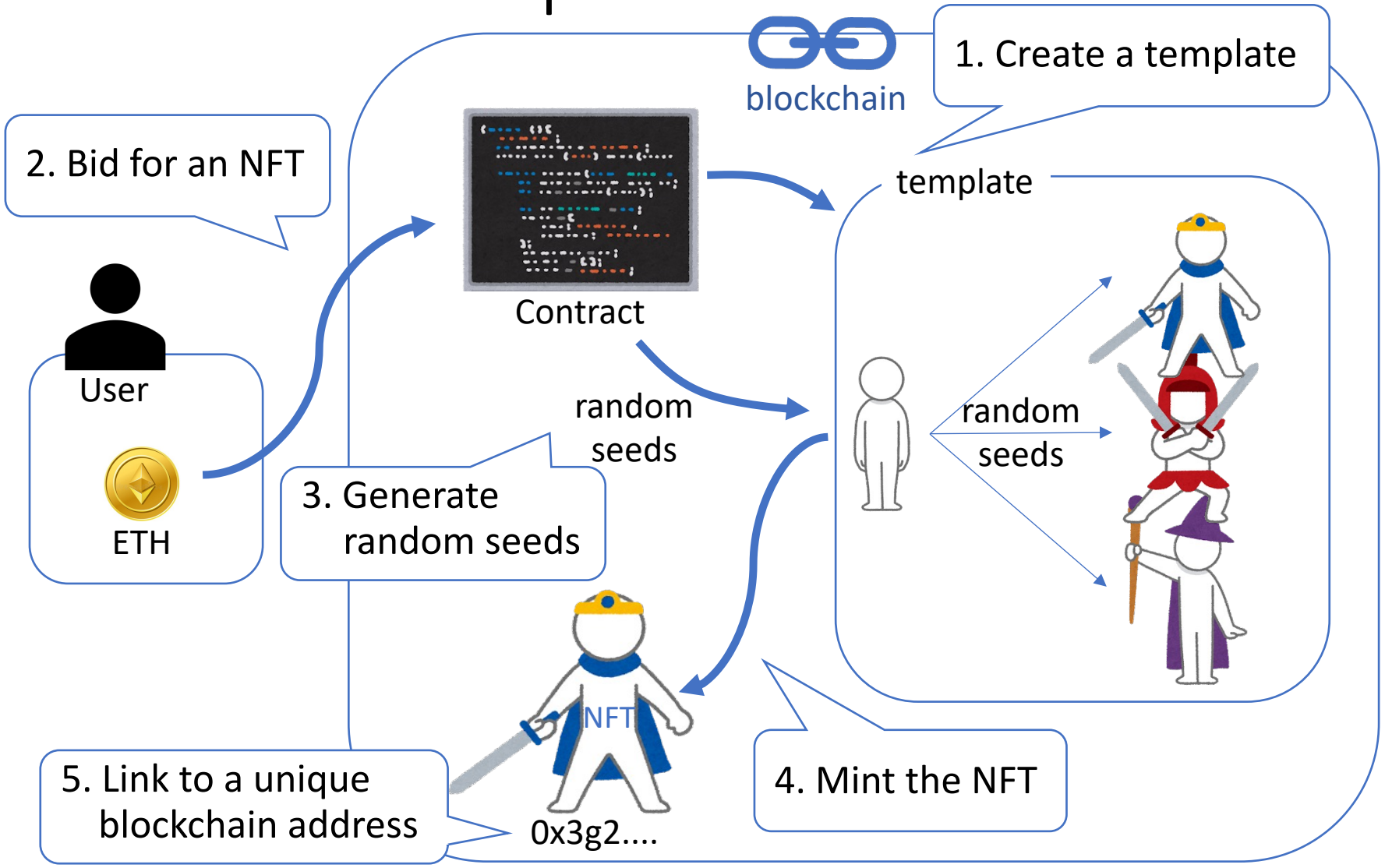
- Top to Bottom
  - digital arts, etc.

- Bottom to Top
  - characters and items in the game, etc.

How to create NFTs Top to Bottom



How to create NFTs Bottom to Top



Challenges

- Usability Challenges
- Security and Privacy Issues
- Governance Consideration
- Extensibility Issues

Conclusion

- Ethereum Smart Contract is new technology.
  - There are various vulnerabilities.
    → Developers must be attention to vulnerabilities.

- Example of NFT
  - Digital arts, Game characters
  - Graduation certificate

→ Believe that widely use

References

- Andreas M. Antonopoulos, Dr. Gavin Wood- ”Mastering Ethereum”

- Q. Wang, et al., “Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges”

Appendix

operating principles of smart contract

- Smart contracts are written in high-level language and compiled to be understood by EVMs

(Basically, every Ethereum participant has EVM)

- The EVM bytecode is deployed to BlockChain as special transaction

- The code is available to every participant

operating principles of smart contract2

- Contract exec requester send information about contract, parameter, commition to network
- Miners execute the contract and write the result of contract to BlockChain and get extra reward