Chapter 14 유명한 이더리움 NFT 프로젝트 살펴보기

BAYC
컨트랙트 생성 ~ 민팅

BAYC

지루한 원숭이들의 요트 클럽
(Bored Ape Yache Club)

대표적인 PFP NFT 의 하나

소스코드:
https://etherscan.io/token/0xbc4ca0eda7647a8ab7c2061c2e118a18a936f13d#code

컨트랙트 생성

컨트랙트 이름은 BoredApeYacheClub 이며,
ERC721, Ownable 두 컨트랙트를 상속

```
contract BoredApeYachtClub is ERC721, Ownable {
    using SafeMath for uint256;

    string public BAYC_PROVENANCE = "";

    uint256 public startingIndexBlock;

    uint256 public startingIndex;

    uint256 public constant apePrice = 80000000000000000; //0.08 ETH

    uint public constant maxApePurchase = 20;

    uint256 public MAX_APES;

    bool public saleIsActive = false;

    uint256 public REVEAL_TIMESTAMP;

    constructor(string memory name, string memory symbol, uint256 maxNftSupply, uint256 saleStart) ERC721(name, symbol) {
        MAX_APES = maxNftSupply;
        REVEAL_TIMESTAMP = saleStart + (86400 * 9);
    }
```

컨트랙트 생성

apePrice 는 0.08ETH. 초기 판매 가격과 연관있을 것으로 추정할 수 있음.

```
contract BoredApeYachtClub is ERC721, Ownable {
    using SafeMath for uint256;

    string public BAYC_PROVENANCE = "";

    uint256 public startingIndexBlock;

    uint256 public startingIndex;

    uint256 public constant apePrice = 80000000000000000; //0.08 ETH

    uint public constant maxApePurchase = 20;

    uint256 public MAX_APES;

    bool public saleIsActive = false;

    uint256 public REVEAL_TIMESTAMP;

    constructor(string memory name, string memory symbol, uint256 maxNftSupply, uint256 saleStart) ERC721(name, symbol) {
        MAX_APES = maxNftSupply;
        REVEAL_TIMESTAMP = saleStart + (86400 * 9);
    }
```

컨트랙트 생성

MAX_APES = 최대 NFT 개수 (판매 NFT 개수)

REVEAL_TIMESTAMP = NFT 공개 시점

```
contract BoredApeYachtClub is ERC721, Ownable {
    using SafeMath for uint256;

    string public BAYC_PROVENANCE = "";

    uint256 public startingIndexBlock;

    uint256 public startingIndex;

    uint256 public constant apePrice = 80000000000000000; //0.08 ETH

    uint public constant maxApePurchase = 20;

    uint256 public MAX_APES;

    bool public saleIsActive = false;

    uint256 public REVEAL_TIMESTAMP;

    constructor(string memory name, string memory symbol, uint256 maxNftSupply, uint256 saleStart) ERC721(name, symbol) {
        MAX_APES = maxNftSupply;
        REVEAL_TIMESTAMP = saleStart + (86400 * 9);
    }
```

민팅

mintApe 함수 (민팅 함수)

민팅 함수를 통해 초기 BAYC NFT 판매

saleIsActive 변수가 true 일때만 판매
(saleIsActive 변수는 BAYC 개발자가 활성화 가능)

```
/**
* Mints Bored Apes
*/
function mintApe(uint numberOfTokens) public payable {
    require(saleIsActive, "Sale must be active to mint Ape");
    require(numberOfTokens <= maxApePurchase, "Can only mint 20 tokens at a time");
    require(totalSupply().add(numberOfTokens) <= MAX_APES, "Purchase would exceed max supply of Apes");
    require(apePrice.mul(numberOfTokens) <= msg.value, "Ether value sent is not correct");

    for(uint i = 0; i < numberOfTokens; i++) {
        uint mintIndex = totalSupply();
        if (totalSupply() < MAX_APES) {
            _safeMint(msg.sender, mintIndex);
        }
    }

    // If we haven't set the starting index and this is either 1) the last saleable token or 2) the first token to be sold after
    // the end of pre-sale, set the starting index block
    if (startingIndexBlock == 0 && (totalSupply() == MAX_APES || block.timestamp >= REVEAL_TIMESTAMP)) {
        startingIndexBlock = block.number;
    }
}
```

민팅

함수 한 번 당 최대로 구매할 수 있는 NFT 제한.
트랜잭션 한 번으로 수많은 NFT 를 사들여
독점하는 것을 방지하기 위함.

(단, 완벽하게 막을 수 있는 방법은 아님)

```
/**
* Mints Bored Apes
*/
function mintApe(uint numberOfTokens) public payable {
    require(saleIsActive, "Sale must be active to mint Ape");
    require(numberOfTokens <= maxApePurchase, "Can only mint 20 tokens at a time");
    require(totalSupply().add(numberOfTokens) <= MAX_APES, "Purchase would exceed max supply of Apes");
    require(apePrice.mul(numberOfTokens) <= msg.value, "Ether value sent is not correct");

    for(uint i = 0; i < numberOfTokens; i++) {
        uint mintIndex = totalSupply();
        if (totalSupply() < MAX_APES) {
            _safeMint(msg.sender, mintIndex);
        }
    }

    // If we haven't set the starting index and this is either 1) the last saleable token or 2) the first token to be sold after
    // the end of pre-sale, set the starting index block
    if (startingIndexBlock == 0 && (totalSupply() == MAX_APES || block.timestamp >= REVEAL_TIMESTAMP)) {
        startingIndexBlock = block.number;
    }
}
```

민팅

최대 발행량 제한.

민팅 시 최대 발행량이 초과되지 않도록 함.

```
/**
* Mints Bored Apes
*/
function mintApe(uint numberOfTokens) public payable {
    require(saleIsActive, "Sale must be active to mint Ape");
    require(numberOfTokens <= maxApePurchase, "Can only mint 20 tokens at a time");
    require(totalSupply().add(numberOfTokens) <= MAX_APES, "Purchase would exceed max supply of Apes");
    require(apePrice.mul(numberOfTokens) <= msg.value, "Ether value sent is not correct");

    for(uint i = 0; i < numberOfTokens; i++) {
        uint mintIndex = totalSupply();
        if (totalSupply() < MAX_APES) {
            _safeMint(msg.sender, mintIndex);
        }
    }

    // If we haven't set the starting index and this is either 1) the last saleable token or 2) the first token to be sold after
    // the end of pre-sale, set the starting index block
    if (startingIndexBlock == 0 && (totalSupply() == MAX_APES || block.timestamp >= REVEAL_TIMESTAMP)) {
        startingIndexBlock = block.number;
    }
}
```

민팅

트랜잭션에서 가격 x 구매 개수 이상의 가격을 전송해야 NFT 민팅 가능

초기 BAYC 에서는 NFT 1개 당 0.08ETH 으로 판매함.

```
/**
* Mints Bored Apes
*/
function mintApe(uint numberOfTokens) public payable {
    require(saleIsActive, "Sale must be active to mint Ape");
    require(numberOfTokens <= maxApePurchase, "Can only mint 20 tokens at a time");
    require(totalSupply().add(numberOfTokens) <= MAX_APES, "Purchase would exceed max supply of Apes");
    require(apePrice.mul(numberOfTokens) <= msg.value, "Ether value sent is not correct");

    for(uint i = 0; i < numberOfTokens; i++) {
        uint mintIndex = totalSupply();
        if (totalSupply() < MAX_APES) {
            _safeMint(msg.sender, mintIndex);
        }
    }

    // If we haven't set the starting index and this is either 1) the last saleable token or 2) the first token to be sold after
    // the end of pre-sale, set the starting index block
    if (startingIndexBlock == 0 && (totalSupply() == MAX_APES || block.timestamp >= REVEAL_TIMESTAMP)) {
        startingIndexBlock = block.number;
    }
}
```

민팅

구매 횟수만큼 민팅

최대 발행량을 이미 체크했는데 민팅하면서 또 체크하는 이유?

Reentrancy Attack 을 막기 위함.

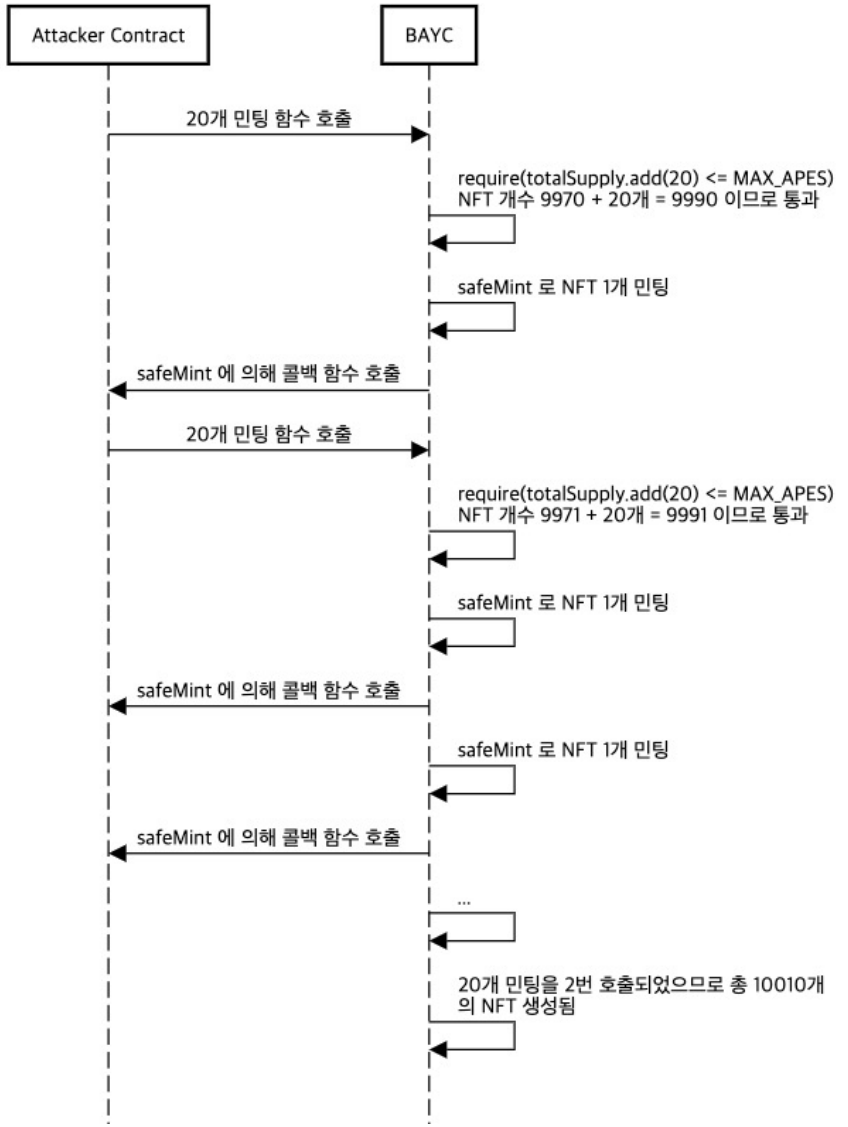
_safeMint 에서 msg.sender 가 Contract 일 경우 콜백 함수 호출. 콜백 함수에서 다시 mintApe 함수를 호출할 경우 최대 발행량 이상으로 발행될 수 있음.

```
/**
* Mints Bored Apes
*/
function mintApe(uint numberOfTokens) public payable {
    require(saleIsActive, "Sale must be active to mint Ape");
    require(numberOfTokens <= maxApePurchase, "Can only mint 20 tokens at a time");
    require(totalSupply().add(numberOfTokens) <= MAX_APES, "Purchase would exceed max supply of Apes");
    require(apePrice.mul(numberOfTokens) <= msg.value, "Ether value sent is not correct");

    for(uint i = 0; i < numberOfTokens; i++) {
        uint mintIndex = totalSupply();
        if (totalSupply() < MAX_APES) {
            _safeMint(msg.sender, mintIndex);
        }
    }

    // If we haven't set the starting index and this is either 1) the last saleable token or 2) the first token to be sold after
    // the end of pre-sale, set the starting index block
    if (startingIndexBlock == 0 && (totalSupply() == MAX_APES || block.timestamp >= REVEAL_TIMESTAMP)) {
        startingIndexBlock = block.number;
    }
}
```

Reentrancy Attack

재진입 공격.

이더 전송, NFT 에서의 safeMint 등 함수 호출 시 msg.sender 가 Contract 일 경우 Contract 에 콜백 함수를 호출하게 됨.
함수 실행이 완료되지 않아 상태 변경이 이루어지지 않은 상태에서 이더 전송, safeMint 등의 함수 호출으로 컨트랙트의 콜백 함수에서 실행되고, 기존 컨트랙트의 함수를 호출하여 재진입하면서 해킹하는 기법

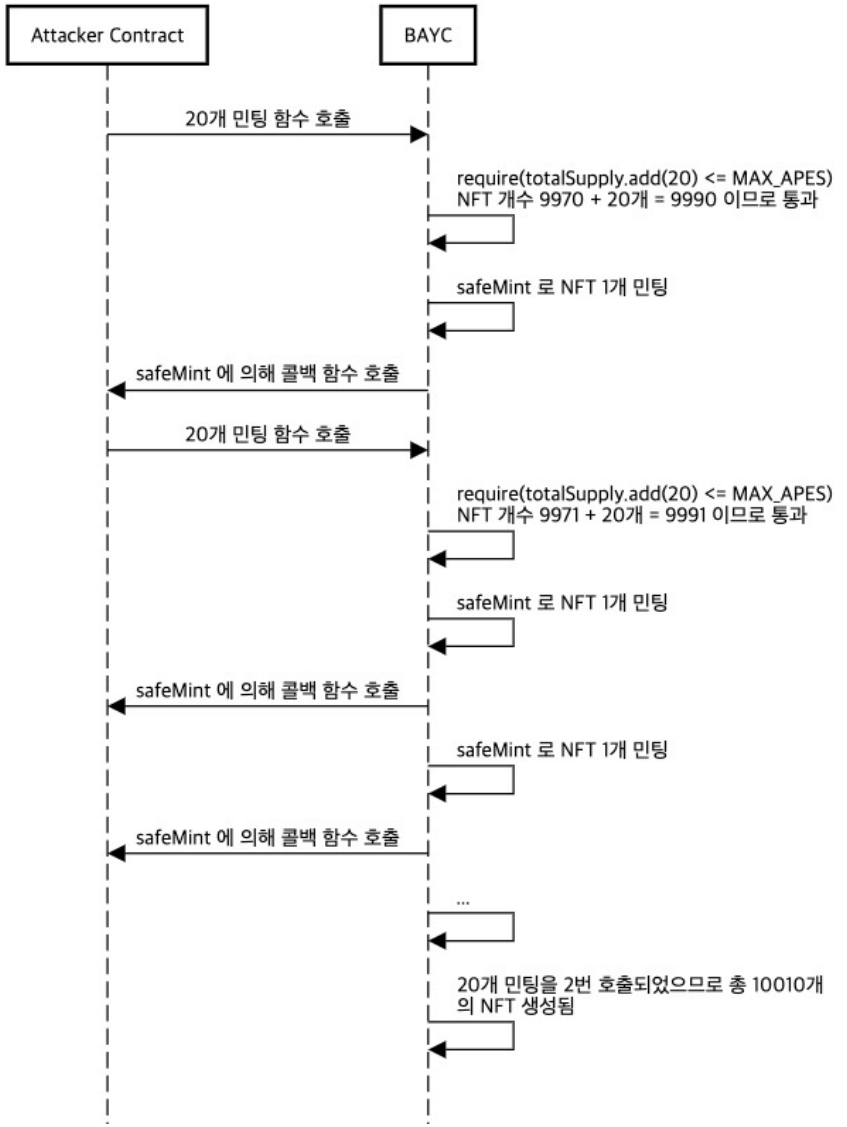


Reentrancy Attack

20개 민팅 함수 호출
safeMint 에 의해 Attacker Contract 에 콜백 함수가 호출되고, Attacker Contract 에서 다시 20 개 민팅 함수 호출 (20개 민팅 2번 요청)

아직 민팅이 완료되지 않은 상태에서 조건문을 체크함으로 최대 발행량 제한 조건문 통과

결과적으로 최대 발행량 초과로 NFT 가 민팅될 수 있는 케이스 발생 가능



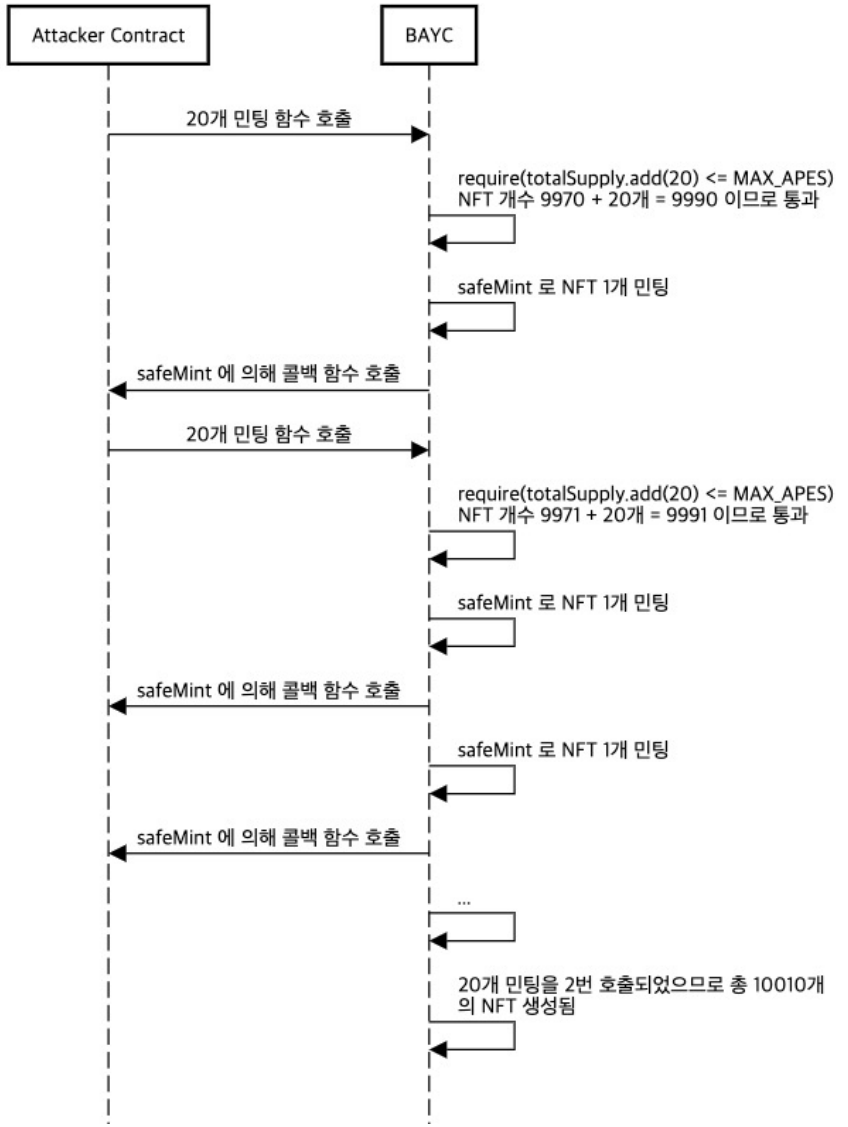
Reentrancy Attack

재진입 공격을 방지하기 위해 Check - Effect - Interaction 패턴 적용

Check : 조건문으로 체크
Effect : 변수 값 수정
Interaction : 컨트랙트로 callback 함수 실행

ex) BAYC 에서는
Check : 최대발행량(totalSupply) 가 10000개 미만인가?
(추가로 민팅이 가능한가)
Effect : NFT 민팅 (NFT 소유권을 위한 _owners 등 변수 수정)
Interaction : 컨트랙트에 callback 함수 실행

→ _safeMint 에서 **Effect 및 Interaction 영역** 수행



민팅

Reveal 시 NFT 를 랜덤으로 드롭하기 위해 블록 넘버 기록

Reveal 시 유저에게 랜덤으로 이미지를 제공하기 위해 이미지를 섞는 과정에서 랜덤 값 필요

랜덤 값을 사용하기 위해 블록 해시 값을 사용

어떤 블록의 해시값을 사용할건지 결정하는 것. (마지막 10000번째 NFT 판매 완료 시 블록 넘버를 기록)

```
/**
* Mints Bored Apes
*/
function mintApe(uint numberOfTokens) public payable {
    require(saleIsActive, "Sale must be active to mint Ape");
    require(numberOfTokens <= maxApePurchase, "Can only mint 20 tokens at a time");
    require(totalSupply().add(numberOfTokens) <= MAX_APES, "Purchase would exceed max supply of Apes");
    require(apePrice.mul(numberOfTokens) <= msg.value, "Ether value sent is not correct");

    for(uint i = 0; i < numberOfTokens; i++) {
        uint mintIndex = totalSupply();
        if (totalSupply() < MAX_APES) {
            _safeMint(msg.sender, mintIndex);
        }
    }

    // If we haven't set the starting index and this is either 1) the last saleable token or 2) the first token to be sold after
    // the end of pre-sale, set the starting index block
    if (startingIndexBlock == 0 && (totalSupply() == MAX_APES || block.timestamp >= REVEAL_TIMESTAMP)) {
        startingIndexBlock = block.number;
    }
}
```