18 DEC 2018

Piction Network

SMART CONTRACT AUDIT REPORT

HAECHI LABS

01. INTRODUCTION

본 보고서는 Piction Network 팀이 제작한 토큰 스마트 컨트랙트의 보안을 감사하기 위해 작성되었습니다. “HAECHI LABS” 팀에서는 Piction Network 팀이 제작한 스마트 컨트랙트의 구현 및 설계가 공개된 자료에 명시한 것처럼 잘 구현이 되어있고, 보안상 안전한지에 중점을 맞춰 감사를 진행했습니다.

Audit 에 사용된 코드는 https://github.com/piction-protocol/piction-ico/tree/master/contracts/token에서 찾아볼 수 있습니다. Audit 에 사용된 코드의 마지막 커밋은 “346af66906b101a4359ac6506acc5f651fc439e2” 입니다.

02. CONTRACTS SUBJECT TO AUDIT

- ContractReceiver.sol
- CustomToken.sol
- ExtendsOwnable.sol
- PXL.sol

03. ABOUT HAECHI LABS

“HAECHI LABS” 는 기술을 통해 건강한 블록체인 생태계에 기여하자는 비전을 가지고 있습니다. “HAECHI LABS”는 스마트 컨트랙트의 보안뿐만 아니라 블록체인 기술에 깊이 있는 연구를 진행하고 있습니다. The DAO, Parity Multisig Wallet, SmartMesh(ERC20) 해킹 사건과 같이 스마트 컨트랙트의 보안 취약점을 이용한 사건들이 지속해서 발생하고 있습니다.

“HAECHI LABS”는 이러한 보안 사고를 예방하기 위해 안전한 스마트 컨트랙트 설계와 구현 및 보안 감사에 최선을 다합니다. 고객사가 목적에 맞는 안전한 스마트 컨트랙트를 구현하고 운영 시 발생하는 가스비를 최적화할 수 있도록 스마트 컨트랙트 관련 서비스를 제공합니다. “HAECHI LABS”는 스타트업에서 다년간 Software Engineer로 개발을 한 경험과 블록체인 연구 경험을 지닌 연구진으로 구성되어 있습니다.

04. SUMMARY

Piction Network는 openzeppelin-solidity 라이브러리를 활용하여

- 소각(burn)
- 토큰 발행(mint)
- 토큰 전송 잠금(lock)

이 가능한 토큰 컨트랙트를 구현하였습니다.

또한 ExtendsOwnable 컨트랙트내에 여러 명의 owner를 둘 수 있는 권한과 관련된 로직을 구현하였습니다.

PXL.sol에서는 approveAndCall이란 함수를 자체적으로 구현했습니다. 해당 함수의 주 목적은 Contract Account에게 PXL 토큰의 사용 권리를 넘겨주는 작업을 간단히 하기 위함입니다. 기존 ERC20의 approve 함수를 통한 방식에서는

1. 컨트랙트에게 토큰 사용 권한 부여(approve 함수)
2. 컨트랙트가 권한을 이용해 사용자의 토큰을 사용(transferFrom 함수)

두 단계의 절차(두개의 트랜잭션)를 거쳤습니다. 이는 단순히 토큰을 컨트랙트 계좌로 전송하는 방식으로는 컨트랙트가 토큰을 받았는지 알 수 없어 이후의 로직을 실행할 수 없기 때문입니다.

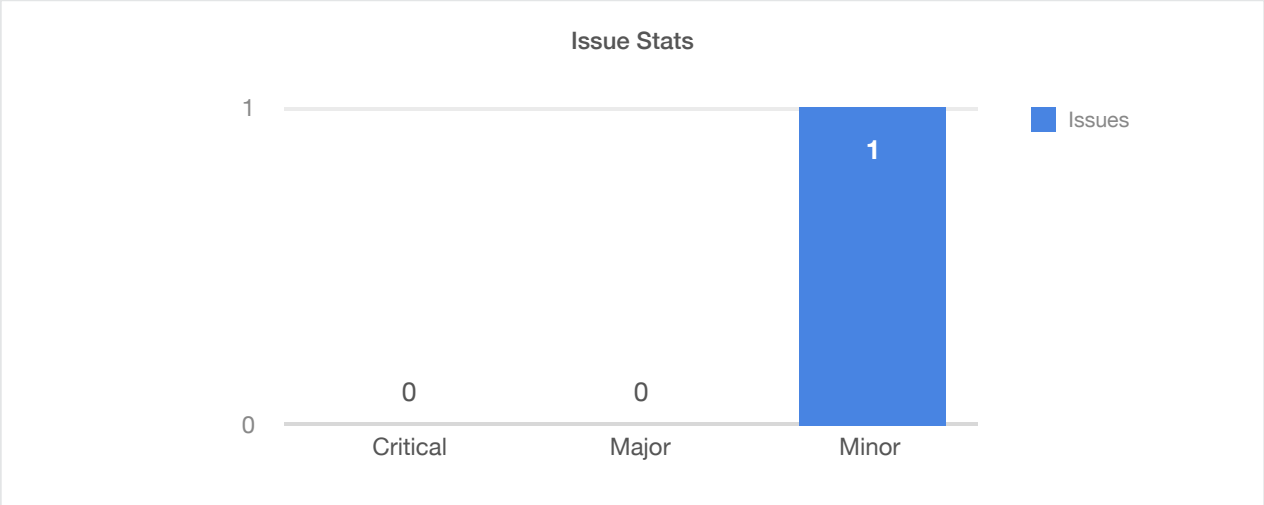
approveAndCall 함수는 해당 함수 실행과 동시에 수신자가 컨트랙트일 경우 해당 컨트랙트의 receiveApproval 함수를 호출합니다. 수신 컨트랙트는 ContractReceiver.sol을 상속 받아 receiveApproval 함수를 구현하여 token을 받았을때 즉각적으로 원하는 로직을 작동 할 수 있습니다.

HAECHI LABS는 Piction Network의 권한 관리 컨트랙트에서 몇 가지 개선사항을 발견했습니다.

05. ISSUES FOUND

발견된 이슈는 중요도 차이에 따라 Critical, Major, Minor로 나누어집니다. Critical 이슈는 광범위한 사용자가 피해를 볼 수 있는 치명적인 보안 결점으로 반드시 해결해야 하는 사항입니다. Major 이슈는 보안상에 문제가 있거나 의도와 다른 구현으로 수정이 필요한 사항입니다. Minor 이슈는 잠재적으로 문제를 발생시킬 수 있으므로 수정이 요구되는 사항입니다. “HAECHI LABS”는 Piction Network 팀이 발견된 모든 이슈에 대하여 개선하는 것을 권장합니다. 이어지는 이슈 설명에서는 코드를 세부적으로 지칭하기 위해서 {파일 이름}:{줄 번호} 포맷을 사용합니다. 예를 들면, PXL.sol:20은 PXL.sol 파일의 20번째 줄을 지칭합니다.

Issue Stats



[그림 1] Issue Stats

1) *ExtendsOwnable*에서 잘못 추가된 owner를 삭제할 수 없습니다 *-Minor*

```solidity
pragma solidity ^0.4.24;

contract ExtendsOwnable {

    mapping(address => bool) owners;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
    event OwnershipExtended(address indexed host, address indexed guest);

    modifier onlyOwner() {
        require(owners[msg.sender]);
        _;
    }

    constructor() public {
        owners[msg.sender] = true;
    }

    function addOwner(address guest) public onlyOwner {
        require(guest != address(0));
        owners[guest] = true;
        emit OwnershipExtended(msg.sender, guest);
    }

    function transferOwnership(address newOwner) public onlyOwner {
        require(newOwner != address(0));
        owners[newOwner] = true;
        delete owners[msg.sender];
        emit OwnershipTransferred(msg.sender, newOwner);
    }
}
```

ExtendsOwnable.sol - https://github.com/piction-protocol/piction-ico/blob/master/contracts/utils/ExtendsOwnable.sol

PROBLEM STATEMENT

PXL의 주요 권한 관리 로직이 작성된 ExtendsOwnable의 경우, owner를 추가(addOwner)하거나 이전(transferOwnership)하는 로직만 구현되어 있습니다. 만약 잘못된 address를 owner로 추가했을 경우, 해당 address를 삭제하는 로직이 구현되어 있지 않습니다. unlock 시기가 되기 전(e.g. 상장 전), 악의적인 owner는 이를 이용하여 원하는 주소들을 owner에 추가해 마음대로 토큰을 유동화시킬 수 있습니다.

EXPLOIT SCENARIO

a. unlock 시기가 되기 전(e.g. 상장 전), A를 owner로 추가합니다.

b. A는 B를 관리자로 등록하였어야 했지만 실수로 C를 관리자로 등록합니다.

c. 하지만 관리자를 등록 해제하는 함수가 없으므로 C는 권한이 없음에도 관리자로서 역할을 유지합니다.

RECOMMENDATION

잘못 추가한 owner를 삭제할 수 있는 removeOwner를 도입하시길 추천합니다. 이 때 Piction Network팀의 address를 누구도 지울 수 없는 super owner로 지정합니다. 다른 owner들에 대한 remove 기능은 super owner에게만 허락할 수도 있고 모두에게 허락할 수도 있습니다.

EXCEPTIONAL CIRCUMSTANCES

만일 Piction Network에서 지향하는 정책 방향이 remove 기능이 없는 방향이라면 해당 사항은 수정을 필요로 하지 않습니다.

06. TIPS

*ExtendsOwnable*에서 *owners* 를 public 변수로 설정하시기 바랍니다

```
3   contract ExtendsOwnable {
4
5       mapping(address => bool) owners;
```

ExtendsOwnable.sol:5 - https://github.com/piction-protocol/piction-ico/blob/master/contracts/utils/ExtendsOwnable.sol#L5

PROBLEM STATEMENT

ExtendsOwnable.sol:5의 *owners*는 토큰 *mint*와 *burn* 권한을 가지고 있으며 lock 상태일 때 전송할 수 있는 주소들을 나타냅니다. 즉, *owners*는 현재 어떤 주소가 그러한 권한을 가지고 있는지 알려주는 중요한 역할을 합니다. 그러나 해당 변수는 현재 public으로 선언되어 있지 않고 별도의 getter 함수 역시 구현되어있지 않아 값을 읽기가 번거롭습니다. 해당 변수의 내용을 알기 위해서는 과거 발생한 트랜잭션 값을 보고 해석하거나 이벤트를 통해 간접적으로 확인 해야합니다. 만약 해당 변수가 public으로 선언되어 있거나 getter함수가 구현되어 있었다면 간단히 컨트랙트의 함수를 호출하는 것으로 해당 변수의 값을 읽어올 수 있습니다.

RECOMMENDATION

```
1   pragma solidity ^0.4.24;
2
3   contract ExtendsOwnable {
4
5       mapping(address => bool) public owners;
```

*owners*에 *public* modifier를 붙이는 것을 추천해 드립니다.

07. TEST RESULTS

아래 결과는, 보안 감사 대상인 스마트 컨트랙트의 주요 로직을 커버하는 unit test 결과입니다.

```
Contract: ExtendsOwnable
  Control Ownership
    #addOner()
      ✓ should disallow anyone to call function (25880 gas)
      ✓ should disallow to add zero address (28706 gas)
      ✓ should add new owner (136518 gas)
    #transferOwnership()
      ✓ should disallow anyone to call function (25902 gas)
      ✓ should disallow to transfer ownership to zero address (28728 gas)
      ✓ should transfer ownership (208708 gas)

Contract: PXL
  ✓ #name()
  ✓ #symbol()
  ✓ #decimals()
  #totalsupply()
    ✓ should return 0 at first
  #fallback()
    ✓ should not receive ethers (22810 gas)
  #mint()
    ✓ should disallow anyone to mint  (24828 gas)
    ✓ should mint properly (72772 gas)
  #burn()
    ✓ should disallow anyone to burn  (24740 gas)
    ✓ should burn properly (43044 gas)
  Control Transfers
    ✓ should unlock by owners only (72658 gas)
    Before Unlock
      ✓ should be locked at first
      ✓ should allow only owners to call transfer() (82839 gas)
      ✓ should allow only owners to call transfer() (223590 gas)
      ✓ should allow only owners to call approveAndCall() (87831 gas)
    After Unlock
      ✓ should allow anyone to call transfer() (97574 gas)
      ✓ should allow anyone to call transferFrom() (150248 gas)
      #approveAndCall()
        ✓ should disallow calling approveAndCall to zero address (28026 gas)
        ✓ should disallow calling approveAndCall to itself (29394 gas)
        ✓ should disallow to approveAndCall excess value (32185 gas)
        ✓ should act as approve() when receiver is EOA (60419 gas)
        ✓ should call receiver's receiveApproval() if the receiver is contact (98460 gas)

27 passing (17s)
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
token/	100	100	100	100	
ContractReceiver.sol	100	100	100	100	
CustomToken.sol	100	100	100	100	
PXL.sol	100	100	100	100	
token/mock/	100	100	100	100	
ContractReceiverMock.sol	100	100	100	100	
utils/	100	100	100	100	
ExtendsOwnable.sol	100	100	100	100	
All files	100	100	100	100	

[그림 2] Testcase code coverage

Gas					Block limit: 17592186044415 gas
Methods		5 gwei/gas		97872.30 krw/eth	
Contract	Method	Min	Max	Avg	# calls	krw (avg)
ExtendsOwnable	addOwner	-	-	55319	4	27.07
ExtendsOwnable	transferOwnership	-	-	46310	1	22.66
PXL	approve	-	-	45330	3	22.18
PXL	approveAndCall	60419	98460	73192	3	35.82
PXL	burn	-	-	43044	1	21.06
PXL	mint	-	-	72772	1	35.61
PXL	transfer	41287	56566	53399	5	26.13
PXL	transferFrom	48631	48910	48771	2	23.87
PXL	unlock	-	-	47984	1	23.48
Deployments				% of limit	
ContractReceiverMock		-	-	321205	0 %	157.19
ExtendsOwnable		-	-	1123758	0 %	549.92
PXL		-	-	3914211	0 %	1915.46

[그림 3] Ether Gas Report

Smart Contract Security Tool Reports

Symbolic Execution
총 3개의 경고가 있었으나, 모두 false positive로 판단됩니다.

Static Analysis
발견된 경고 혹은 에러가 없습니다.

Logs:
INFO:Slither:/tmp/flat_PXL.sol analyzed (6 contracts), 0 result(s) found

08. CONCLUSION

Piction Network 의 토큰 컨트랙트에서 Minor 이슈 1개가 발견되었습니다. Minor 이슈는 잠재적으로 문제를 발생시킬 수 있으므로 수정이 요구되는 사항입니다. 그리고 보안상으로 크게 문제는 없지만, 수정했을 때 코드의 사용성이나 효율성이 더 좋아질 수 있는 사항을 Tips 항목에 서술하였습니다. “HAECHI LABS”에서는 모든 발견된 이슈에 대해 수정하는 것을 권장합니다.

09. DISCLAIMER

해당 리포트는 투자에 대한 조언, 비즈니스 모델의 적합성, 버그 없이 안전한 코드를 보증하지 않습니다. 해당 리포트는 알려진 기술 문제들에 대한 논의의 목적으로만 사용됩니다. 리포트에 기술된 문제 외에도 이더리움, 솔리디티 상의 결함 등 발견되지 않은 문제들이 있을 수 있습니다. 안전한 스마트 컨트랙트를 작성하기 위해서는 발견된 문제들에 대한 수정과 충분한 테스트가 필요합니다.