18 DEC 2018

Piction Network

SMART CONTRACT AUDIT REPORT

HAECHI LABS

01. INTRODUCTION

This audit report covers the smart contract security of the token smart contract created by the Piction Network team. “HAECHI LABS” audited the smart contract codes produced by the Piction Network team to ensure that the codes were designed for the purpose outlined in the white paper and other published materials, and review whether the codes are secure.

The codes used for the audit are given by the Piction Network team. The code used for this audit can be found in the Piction Network’s Github repository (https://github.com/piction-protocol/piction-ico/tree/master/contracts/token) and the final commit of the code used for the audit is “346af66906b101a4359ac6506acc5f651fc439e2”

02. CONTRACTS SUBJECT TO AUDIT

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

03. ABOUT HAECHI LABS

"HAECHI LABS" provides smart contract security audit, DApp development and enterprise blockchain solutions to easily integrate blockchain technology into daily life. The "HAECHI LABS" team consists of blockchain and smart contract developers with niche skills who conduct in-depth research on smart contract security as there have been many incidents caused by smart contract vulnerabilities.

Moreover, "HAECHI LABS" is committed to providing the highest quality smart contract audit. Smart contract experts of "HAECHI LABS" conduct smart contract audit by using static analysis based on symbolic execution, examining all possible attack scenarios via unit testing and proposing the gas optimization by designing a better architecture in terms of tech consulting.

04. SUMMARY

The Piction Network team used the “openzeppelin-solidity library” to implement token contract which enables

- burn token
- issue token (mint)
- Token transfer lock.

Piction Network team implemented the authority logic to set multiple owners in the ExtendsOwnable contract. Also, in the PXL.sol, Piction Network team implemented its own function called approveAndCall. The main purpose of this function is to simplify the task of passing the authority when using the PXL token to the Contract Account.

In the previous ERC20’ method through approve function, it usually follows the following steps (two transactions) which are :

1. Grant the authority of using tokens to the contract (approve function)
2. The contract uses its authority to use user's tokens (transferFrom function).

This is because by simply transferring tokens to the Contract Account, it is difficult to notice whether the contract really received tokens or not which eventually prevents one from executing the subsequent logic.

Meanwhile, the approveAndCall function calls the receiveApproval function of the corresponding contract simultaneously with the function execution if the receiver is a contract.

The receiving contract can inherit the ContractReceiver.sol and implement the receiveApproval function. This enables calling the desired function instantly when token is received.

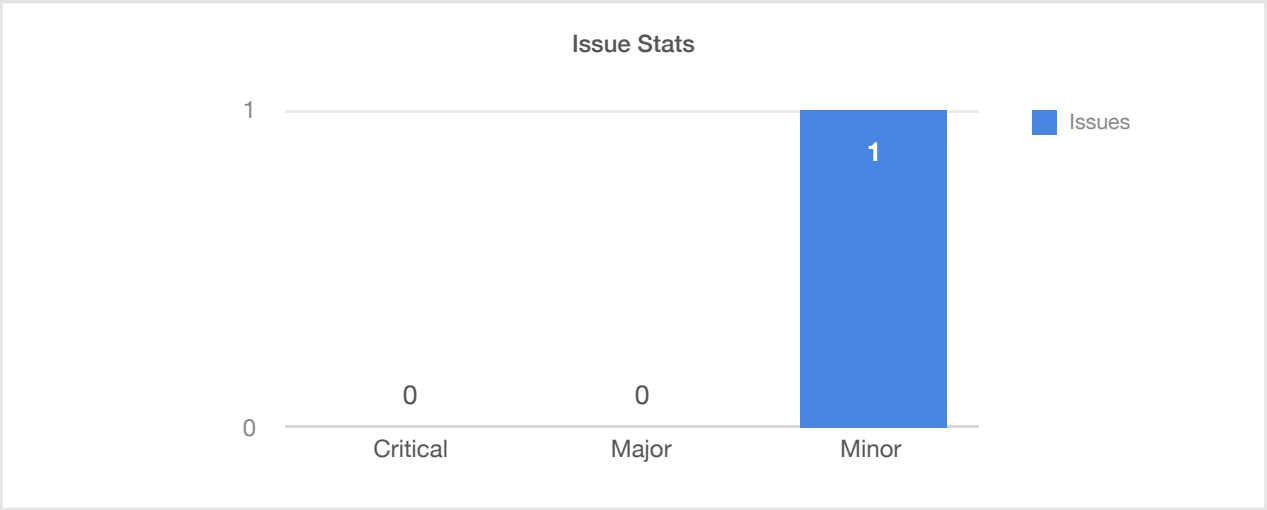
HAECHI LABS has found some improvements in the authority management contract of the Piction Network.

05. ISSUES FOUND

The found issues are classified into Critical, Major, Minor, and Informative depending on their importance. Critical issues are fatal flaws in security which must be addressed since it may damage a wide range of users. Major issues are either security issues or issues that require modifications due to unintended implementation. Minor issues require modifications because it can potentially cause problems. Lastly, informative issues are recommended modifications to improve the usability and efficiency of codes.

HAECHI LABS recommends the Piction Network team to improve all the found issues. We use the format, {file name}:{line number} in the following issue explanations to indicate code in detail. For example, PXL.sol:20 refers to line 20 of the PXL.sol file.

Issue Stats



[Image 1] Issue Stats

1) Unable to delete owner from *ExtendsOwnable* even when the owner was added unintentionally. *-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

For ExtendsOwnable where the PXL's authority management logic is implemented, the only implemented logic is to add owner(addOwner) or transfer the ownership(transferOwnership).

If the wrong address was added as the owner, then it is not reversible since the logic to delete wrong address is not implemented. Until the time of unlock (e.g. before the listing), the malicious owner can abuse this by adding desired addresses to the owner and freeze the tokens at will.

EXPLOIT SCENARIO

a. Add A as owner before the period of unlock (e.g. before the listing).
b. A should have registered B as an owner, but inadvertently register C as an owner
c. However, because there is no function to revoke ownerships, C will still act as an owner, even if it is not.

RECOMMENDATION

It is recommended to implement a removeOwner function which can revoke the ownership. At this point, the address of the Piction Network team should be designated as the super owner which can not be deleted by anyone. The remove function for other owners is not only limited to the super owner only so it could be granted for other owners as well.

EXCEPTIONAL CIRCUMSTANCES

Unless not having the remove function is the intention of the Piction Network team, the found issue does not necessarily have to be addressed.

06. TIPS

In the *ExtendsOwnable*, set *owners* as public variables in terms of setting

```
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

The *owners* in the ExtendsOwnable.sol:5 have permissions to *mint* and *burn* tokens and indicate the addresses that can be reached for the transfer even in lock state. In other words, *owners* play a significant role in indicating which addresses have such privileges. However, currently the variable is not declared as public and the getter function is yet implemented so it is inconvenient in reading values.

In order to know the contents of these variables, one must trace back the historical transaction values or indirectly interpret through events. If the variable is declared as public or the getter function was implemented, then you can read the value of the variable by simply calling the function of the contract.

RECOMMENDATION

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

It is recommended to add a *public* modifier to the *owners*.

07. TEST RESULTS

The results below are the unit test results covering the main logic of the smart contract subject to the security audit.

```
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	

[Image 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

[Image 3] Ether Gas Report

Smart Contract Security Tool Reports

Symbolic Execution

There were a total of three warnings, all of which are false positives.

Static Analysis

There are no warnings or errors found

Logs:

INFO:Slither:tmpflat_PXL.sol analyzed (6 contracts), 0 result(s) found

08. CONCLUSION

One Minor issue has been found in the token contract of the Piction Network. Minor issues require modifications because it can potentially cause problems. There are no major security issues but still "HAECHI LABS" recommends Piction Network to address all the found issues which can improve the usability and efficiency of the code.

09. DISCLAIMER

The findings of this report is not exhaustive and is limited to the current understanding of known security patterns. This report only discusses known technical issues and does not include any investment advice, the profitability of the business model nor any other issues outside the technical realm. Regardless of the problems described in this report, there may be unknown problems and defects in the Ethereum or the solidity language.In order to create a secure smart contract, you must fix the problems found in this audit and conduct adequate testing.