CRYPTOCURRENCY (HTTPS://BLOG.COINFABRIK.COM/CATEGORY/CRYPTOCURRENCY/)

Security Audit for Patientory (PTOY) Token ICO

Pablo Yabo (https://blog.coinfabrik.com/author/pyabo/)

June 13, 2017 (https://blog.coinfabrik.com/cryptocurrency/security-audit-crowdsale-token-ptoy-token-ico/)

CoinFabrik smart contract audit (http://www.coinfabrik.com)'s team was hired to audit contracts written by TokenMarket for the PTOY Token ICO. The result of this security review is reflected in this document.

Contents

Audited Files

The contracts we audited are hosted at Github repository:

MintableToken.sol
(https://github.com/TokenMarketNet/ico/blob/master/contracts/MintableToken.sol)
CrowdsaleToken.sol
(https://github.com/TokenMarketNet/ico/blob/master/contracts/CrowdsaleToken.sol)
ReleasableToken.sol
(https://github.com/TokenMarketNet/ico/blob/master/contracts/ReleasableToken.sol)
UpgradeableToken.sol
(https://github.com/TokenMarketNet/ico/blob/master/contracts/UpgradeableToken.sol)

Commit hash f968cffe1abf4a3d96d6705ec1befd6cfec13ae3.

Vulnerabilities Found

Short Address Attack

The version reviewed of contract StandardToken which implements transferFrom and approve are not protected against the recently found vulnerability “ERC20 Short Address Attack” (http://vessenes.com/the-erc20-short-address-attack-explained/ (http://vessenes.com/the-erc20-short-address-attack-explained/)).

We should note that the method transfer is protected for this attack.

They should verify the payload size that matches the size of the function parameters:

```
function transferFrom(address _from, address _to, uint _value) returns (bool success) {
 uint _allowance = allowed[_from][msg.sender];

 // Check is not needed because safeSub(_allowance, _value) will already throw if this conditi
 // if (_value > _allowance) throw;

 balances[_to] = safeAdd(balances[_to], _value);
 balances[_from] = safeSub(balances[_from], _value);
 allowed[_from][msg.sender] = safeSub(_allowance, _value);
 Transfer(_from, _to, _value);
 return true;
}

function approve(address _spender, uint _value) returns (bool success) {

 // To change the approve amount you first have to reduce the addresses`
 // allowance to zero by calling `approve(_spender, 0)` if it is not
 // already 0 to mitigate the race condition described here:
 // https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
 if ((_value != 0) && (allowed[msg.sender][_spender] != 0)) throw;

 allowed[msg.sender][_spender] = _value;
 Approval(msg.sender, _spender, _value);
  return true;
}
```

Should add onlyPayloadSize modifier to transferFrom and approve functions:

```
function transferFrom(address _from, address _to, uint _value) onlyPayloadSize(3*32) returns (b
    uint _allowance = allowed[_from][msg.sender];

    // Check is not needed because safeSub(_allowance, _value) will already throw if this condi
    // if (_value > _allowance) throw;

    balances[_to] = safeAdd(balances[_to], _value);
    balances[_from] = safeSub(balances[_from], _value);
    allowed[_from][msg.sender] = safeSub(_allowance, _value);
    Transfer(_from, _to, _value);
    return true;
}
function approve(address _spender, uint _value) onlyPayloadSize(2*32) returns (bool success) {

    // To change the approve amount you first have to reduce the addresses`
    //  allowance to zero by calling `approve(_spender, 0)` if it is not
    //  already 0 to mitigate the race condition described here:
    //  https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
    if ((_value != 0) && (allowed[msg.sender][_spender] != 0)) throw;

    allowed[msg.sender][_spender] = _value;
    Approval(msg.sender, _spender, _value);
    return true;
}
```

Refund tokens

The CrowdsaleToken does not provide a method to refund received tokens. There are cases when a user by mistake has sent token to a contract address. If the contract does not provide a method to recover those tokens, those tokens are held indefinitely by the contract, and cannot be retrieved and are effectively lost.

The following implementation allows the owner to return tokens to a specified address:

```
Event ReclaimTokens(address indexed token, address indexed to, uint value);

function reclaimTokens(address _token, address _to) onlyOwner {
  ERC20 token = ERC20(_token);
  uint balance = token.balanceOf(this);
  if (balance > 0) {
    token.transfer(_to, balance);
    ReclaimTokens(_token, _to, balance);
  }
}
```

StandardToken.sol

Functions transfer, transferFrom and approve return a boolean value. They use safeAdd and safeSub which throws on error conditions. This is inconsistent with the boolean return value, they should return false on failure. Also, ReleasableToken.sol uses modifier canTransfer which throws on error.

Solidity Version

MintableToken.sol v0.4.6
CrowdsaleToken.sol v0.4.8
ReleasableToken.sol v0.4.8
UpgradeableToken.sol v0.4.8

We recommend using v0.4.11 or above since these versions are vulnerable to ConstantOptimizerSubtraction (https://blog.ethereum.org/2017/05/03/solidity-optimizer-bug/) (low-severity)

Duplicated code (minor)

There are two implementations of secure mathematical operations, one in the SafeMath contract and another in the SafeMathLib library.

Having duplicated code perform the same operation it increases the cost of the deployment of the contract unnecessarily. Duplicated code might cause subtle bugs that are hard to debug and from an engineering perspective it increases the burden of testing and reviewing.

In this contract the functionality is almost identical and should not cause problems, beyond an increase in deployment costs.

Implemented improvements

Protection for double spend on approval

Tokens that implement the standard ERC20 are subject to a double spend attack when the user A approves a limit to user B by mistake, and tries to fix calling again with a new limit. If the user B retrieves the tokens between calls it will be allowed another extraction.

The contract code provides a protection against this attack. To modify an address extraction limit it first should be reset to zero. It also provides two methods addApproval and subApproval, that are not part of the ERC20 standard, that provide functionality to alter the limit without the risk of the double spend attack.

Short address attack

The methods transfer, addApproval, and subApproval are reasonably protected against the double spend attack.

```
function transfer(address _to, uint _value) onlyPayloadSize(2 * 32) returns (bool success) {
..

function addApproval(address _spender, uint _addedValue)
onlyPayloadSize(2 * 32)
returns (bool success) {
..

function subApproval(address _spender, uint _subtractedValue)
onlyPayloadSize(2 * 32)

returns (bool success) {
..
```

Do you want to know what is Coinfabrik Auditing Process?

Check our A-Z Smart Contract Audit Guide (https://blog.coinfabrik.com/smart-contract-audits-ultimate-security-guide/) or you could request a quote (https://www.coinfabrik.com#contact_us) for your project.

Related Posts

(https://blog.coinfabrik.com/smart-contracts/tetha-token-sale-security-audit/)

Theta Token Sale Security Audit (https://blog.coinfabrik.com/smart-contracts/tetha-token-sale-security-audit/)

Coinfabrik was asked to audit the contracts for the Theta Token sale. In the first…

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/flixxo-token-sale-audit/)

Flixxo Token Sale Security Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/flixxo-token-sale-audit/)

Coinfabrik's smart contracts auditing team was asked to audit the contracts for the Flixxo Token…

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/bricks4us-token-smart-contract-security-audit/)

Bricks4US Token Smart Contract Security Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/bricks4us-token-smart-contract-security-audit/)

CoinFabrik was asked to audit the contracts for the Bricks4Us token sale. Firstly, we will…

Tags:

- audit (https://blog.coinfabrik.com/tag/audit/)
- Patientory (https://blog.coinfabrik.com/tag/patientory/)
- PTOY (https://blog.coinfabrik.com/tag/ptoy/)
- Token (https://blog.coinfabrik.com/tag/token/)

SHARE ON

(https://www.facebook.com/sharer/sharer.php?u=https://blog.coinfabrik.com/cryptocurrency/security-a
token-ico/)
(https://twitter.com/intent/tweet?text=Security%20Audit%20for%20Patientory%20(PTOY)%20Token%20ICO&url=https://blog.coinfabrik.con
audit-crowdsale-token-ptoy-token-ico/)
(https://pinterest.com/pin/create/button/?url=&media=https://blog.coinfabrik.com/wp-content/uploads/
Patientory.png&description=Security+Audit+for+Patientory+%28PTOY%29+Token+ICO)
(https://www.linkedin.com/shareArticle?mini=true&url=https://blog.coinfabrik.com/cryptocurrency/securi
ptoy-token-ico/&title=Security%20Audit%20for%20Patientory%20(PTOY)%20Token%20ICO&source=CoinF

NEXT ARTICLE →

Security Auditing: Beware of Duplicated Storage in Solidity Smart Contract Development (https://blog.coinfabrik.com/smart-contracts/security-auditing-beware-duplicated-storage-solidity-smart-contract-development/)

← PREVIOUS ARTICLE

Privacy in Cryptocurrencies: An Overview (https://blog.coinfabrik.com/cryptocurrency/privacy-cryptocurrencies-

overview/)

You may also like

(https://blog.coinfabrik.com/smart-contracts/magic-bridge-audit/)

Magic Bridge Audit (https://blog.coinfabrik.com/smart-contracts/magic-bridge-audit/)

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/mintingfactoryv2-baseupgradablemarketplace-kodav3upgradablegatedmarketplace/)

2 months ago

Smart Contract Audit (https://blog.coinfabrik.com/category/smart-contracts/smart-contract-audit-smart-contracts/)

MintingFactoryV2, BaseUpgradableMarketplace & KODAV3UpgradableGatedMarketplace (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/mintingfactoryv2-baseupgradablemarketplace-kodav3upgradablegatedmarketplace/)

(https://github.com/coinfabrik)