(https://blog.coinfabrik.com/)

AUDITORIA SMART CONTRACTS (HTTPS://BLOG.COINFABRIK.COM/CATEGORY/AUDITORIA-SMART-CONTRACTS/)

Bitpara Smart Contract Audit

Mariana Soffer (https://blog.coinfabrik.com/author/mariana-soffel/)

December 5, 2019 (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/bitpara-smart-contract-audit/)

Contents

Introduction

CoinFabrik was asked to audit the contracts for the Bitpara project. Firstly, we will provide a summary of our discoveries and secondly, we will show the details of our findings.

Summary

The contracts audited are from the Bitpara repository at https://github.com/sottile27/bitpara (https://github.com/sottile27/bitpara). The audit is based on the commit b584216d4d0261af9afb50a4a556f693f1763ac5 and updated to reflect changes at commit ffd8e4e5d3b274aa481949434eefa51ec1e5a078.

The audited contracts are:

- **bitpara.sol**

  An ERC20 token that allows the owner to charge a configurable small fee.

The following analyses were performed:

- Misuse of the different call methods: call.value(), send() and transfer().
- Integer rounding errors, overflow, underflow and related usage of SafeMath functions.
- Old compiler version pragmas.
- Race conditions such as reentrancy attacks or front running.
- Misuse of block timestamps, assuming anything other than them being strictly increasing.
- Contract softlocking attacks (DoS).
- Potential gas cost of functions being over the gas limit.
- Missing function qualifiers and their misuse.
- Fallback functions with a higher gas cost than the one that a transfer or send call allows.
- Fraudulent or erroneous code.
- Code and contract interaction complexity.
- Wrong or missing error handling.
- Overuse of transfers in a single transaction instead of using withdrawal patterns.
- Insufficient analysis of the function input requirements.

Severity Classification

The security risk categories are evaluated according to the following classification:

- **Critical:** The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for clients and users.
- **Medium:** The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
- **Minor:** The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
- **Enhancement:** The issue does not pose an immediate threat to continued operation or usage, but is relevant for security best practices, software engineering best practices, or defensive redundancy.

Detailed findings (https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

Critical severity

No issues have been found.

Medium severity

Solidity version is too old

The contract is targeting Solidity compiler version 0.4.18.

```
pragma solidity ^0.4.18;
```

That version was released in October 2017, since then the compiler has several releases fixing numerous bugs. We didn't find any specific vulnerability that might affect the contract.

We recommend to use a more recent version of the compiler. The latest version compatible with source code developed for solidity 0.4 is 0.4.26 which was released in April 2019. Nevertheless, we recommend to consider updating to 0.5 which has improved the language and supports new opcodes released in recent forks like Constantinople and Istanbul.

Minor severity

Owner can make arbitrary transfers

The function *transferToOwner* allows the owner to make a transfer from any user to himself.

```
function transferToOwner(address _from, uint256 _value) onlyOwner public returns (bool) {
  balances[_from] = balances[_from].sub(_value);
  balances[owner] = balances[owner].add(_value);
  Transfer(_from, owner, _value);
  return true;
}
```

The owner can already achieve a similar result by banning a user, burning his balance and minting new tokens. Since the account is banned the balance cannot be used anymore and burning it is reasonable. In the case of *transferToOwner* there is no explicit reason why the balance will be removed from the user.

We recommend to document under which circumstances the owner is allowed to arbitrarily remove balance from a user.

Solidity version is not fixed

The code allows compilation with source code compatible versions of solidity above 0.4.18. New versions sometimes introduce features and bug fixes that might change the expected behavior.

We recommend to always use a fixed version and only upgrade to a new version after testing it didn't cause any regresion.


```
pragma solidity 0.4.26;
```

Use of deprecated constant modifier

Several functions are using the *constant* modifier which was deprecated by the developers of the solidity compiler.

```
  function totalSupply() public constant returns (uint256) {
  function balanceOf(address _owner) public constant returns (uint256 balance) {
  function allowance(address _owner, address _spender) public constant returns (uint256 rema
ining) {
  function getBanStatus(address _unclear) external constant returns (bool) {
  function getOwner() external constant returns (address) {
```

We recommend using the *view* modifiers instead. From version 0.5 of Solidity the modifier is enforced using new opcode STATICCALL making contracts more secure.

The contract was update to use view modifier at commit ffd8e4e5d3b274aa481949434eefa51ec1e5a078.

Enhancements

Unnecessary extra checks

The function *transfer* in BasicToken, *transferFrom* in StandardToken and *burn* function in BurnableToken has redundant checks. First *_values* is checked against the available balance with *require*, and again when using SafeMath to do the subtraction operation.

```
  function transfer(address _to, uint256 _value) public returns (bool) {
    ..
    require(_value <= balances[msg.sender]);

    balances[msg.sender] = balances[msg.sender].sub(_value);
    ..
```

```
function transferFrom(address _from, address _to, uint256 _value) public returns (bool) {
    ..
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    balances[_from] = balances[_from].sub(_value);
    ..
    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
```

```
function burn(uint256 _value) onlyOwner public returns (bool) {
    ..
    require(_value <= balances[owner]);
    balances[owner] = balances[owner].sub(_value);
    ..
  }
```

We recommend to remove redundant checks since it will save some gas when deploying the contract and when users want to interact with it.

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

The unnecessary check were removed at commit ffd8e4e5d3b274aa481949434eefa51ec1e5a078.

Observations

After upgrading to at least solidity compiler v0.4.26 we recommend to apply the following changes.

- Use *constructor* declaration. In solidity 0.4.18 the constructor is a function with the same name as the contract. This might cause problems because a simple mistake in the name will define the constructor as a normal function leaving the contract unitialized and allowing anyone to call it as a normal function after deployment.
- Use *emit* keyword to generate events. In solidity 0.4.18 the syntax to generate an event is identical to a function call which is confusing and requires the context to correctly understand what the contract is doing.
- Replacing *assert* with *require* and adding a message to them. Some block explorers and wallets might display the error message before transaction allowing the user to correct before submitting.

For example:

```
function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    require(b <= a, "Subtraction overflow");
    return a - b;
}
```

Conclusion

The contracts are simple and straightforward to follow.

We found a medium severity issue which is that the contracts target version 0.4.18 of the solidity compiler released in October 2017. We suggest using version 0.4.26 which is source code compatible with code written for 0.4.

Related Posts

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/money-on-chain-security-audit-iii/)

Money on Chain Security Audit III (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/money-on-chain-security-audit-iii/)

CoinFabrik was asked to audit the contracts for the Money On Chain project. We will…

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/timvi-smart-contract-audit/)

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

CoinFabrik was asked to audit the contracts for the Timvi project. Firstly, we will provide…

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/money-on-chain-gas-cost/)

Money on Chain Gas Cost (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/money-on-chain-gas-cost/)

The purpose of this document is to offer different alternatives to reduce gas cost in…

gas-cost/) (https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/smart-contract-auditing-news/)

Smart Contract Auditing News (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/smart-contract-auditing-news/)

Every month several important smart contract audits are performed by blockchain security companies like us.…

(https://blog.coinfabrik.com/auditoria-smart-contracts/money-on-chain-security-audit-i/)

Money on Chain Security Audit I (https://blog.coinfabrik.com/auditoria-smart-contracts/money-on-chain-security-audit-i/)

This is the second of a series of four audits we performed for MOC: Second…

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

Nahmii Security Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/nahmii-security-audit/)

CoinFabrik was asked to audit the contracts for the Nahmii Token project. Firstly, we will…

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/coinfabrik-audit-process/)

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

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

A security audit is a process in which a client subjects his or her smart…

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/easypool-smart-contract-security-audit-v2/)

EasyPool Smart Contract Security Audit v2 (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/easypool-smart-contract-security-audit-v2/)

CoinFabrik has been hired to audit the EasyPool smart contracts. We start this report writing…

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

Stasis Token Smart Contract Audit (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/stasis-token-smart-contract-audit/)

Coinfabrik has been hired to audit the smart contracts for Stasis sale, the Stable Euro…

contract-

audit/)

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/)

(https://blog.coinfabrik.com/smart-

contracts/etherparty-

token-

smart-

contract-

security-

audit-

coinfabrik/)

EtherParty Smart Contract Security Audit

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

Coinfabrik team has been hired to audit Etherparty smart contracts. Firstly, we will provide a…

(https://blog.coinfabrik.com/smart-

contracts/smart-

contract-

audit-

smart-

contracts/rcn-

smart-

contracts-

audit-

v2/)

RCN Smart Contracts Audit v2

(https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/rcn-smart-contracts-audit-v2/)

The smart contracts that have been audited were taken from the RCN repository at: https://github.com/ripio/rcn-network/tree/v2….

(https://blog.coinfabrik.com/smart-

contracts/smart-

contract-

audit-

smart-

DMToken Token Sale Smart Contract Audit

(https://blog.coinfabrik.com/smart-contracts/smart-contract-

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

(https://blog.coinfabrik.com/) Coinfabrik was hired to audit the contract in terms of its security. First of all,...

Tags:

security (https://blog.coinfabrik.com/tag/security/)

Smart Contract (https://blog.coinfabrik.com/tag/smart-contract/)

smart contract audits (https://blog.coinfabrik.com/tag/smart-contract-audits/)

stable-coins (https://blog.coinfabrik.com/tag/stable-coins/)

SHARE ON

(https://www.facebook.com/sharer/sharer.php?u=https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/bitpara-smart-contract-audit/)
(https://twitter.com/intent/tweet?text=Bitpara%20Smart%20Contract%20Audit&url=https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/bitpara-smart-contract-audit/)
(https://pinterest.com/pin/create/button/?url=&media=https://blog.coinfabrik.com/wp-content/uploads/2019/12/bitpara.jpg&description=Bitpara+Smart+Contract+Audit)
(https://www.linkedin.com/shareArticle?mini=true&url=https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/bitpara-smart-contract-audit/&title=Bitpara%20Smart%20Contract%20Audit&source=CoinFabrik%20Blog)

← PREVIOUS ARTICLE

Money on Chain Security Audit III (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/money-on-chain-security-audit-iii/)

NEXT ARTICLE →

Money on Chain Security Audit IV (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/money-on-chain-security-audit-iv/)

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/)