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

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

Beluga Pay (BBI) Security Audit

Ariel Yabo (https://blog.coinfabrik.com/author/ariel-yabo/)

March 1, 2018 (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/beluga-pay-bbi-security-audit/)

Coinfabrik has been hired to audit the smart contracts which were included in the BBI Token sale. In the first part, we will detail a summary of our discoveries and follow them with the details of our findings.

Contents

Summary

The contracts audited are from the BBI repository at https://gitlab.com/cardedeveloper/contractBBIT/blob/master/bbi.sol (https://gitlab.com/cardedeveloper/contractBBIT/blob/master/bbi.sol).

The smart contract can be found at: https://etherscan.io/address/0x37d40510a2f5bc98aa7a0f7bf4b3453bcfb90ac1

The audited contracts are:

- bbi.sol: BBI Token Sale and Token, inherits **StandardToken**.

These checks 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 things other than them being strictly increasing.
- Contract soft locking attacks (DoS).
- Potential gas cost of functions being over the gas limit.
- Missing function qualifiers or misuse of them.
- Fallback functions with higher gas cost than a what a transfer or send call allows.
- Underhanded 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 coverage of function input requirements.

Detailed findings

Critical severity

Function buyBBITokens can be used to get free tokens

When using this contract you are supposed to call the fallback function in order to buy tokens:

```
function () payable onIcoRunning public {

buyBBITokens(msg.sender, msg.value);

}
```

But you may bypass this function and directly call buyBBITokens:

```
function buyBBITokens(address _buyer, uint256 _value) public {
       // prevent transfer to 0x0 address
       require(_buyer != 0x0);

       // msg value should be more than 0
       require(_value > 0);

       // if not halted
       require(!halted);

       // Now is before ICO end date
       require(now < icoEndDate);

       // total tokens is price (1ETH = 960 tokens) multiplied by the ether value provided
       uint tokens = (SafeMath.mul(_value, 960));

       // total used + tokens should be less than maximum available for sale
       require(SafeMath.add(totalUsed, tokens) < balances[addressICOManager]);

       // Ether raised + new value should be less than the Ether cap
       require(SafeMath.add(etherRaised, _value) < etherCap);

       balances[_buyer] = SafeMath.add( balances[_buyer], tokens);
balances[addressICOManager] = SafeMath.sub(balances[addressICOManager], tokens);
       totalUsed += tokens;
       etherRaised += _value;
       addressETHDeposit.transfer(_value);
     Transfer(this, _buyer, tokens );
}
```


This, this would allow you to assign both parameters, in particular, **_value** can be set just high enough to assign yourself all remaining tokens.

Consider qualifying the function visibility as internal instead, in case you wanted to use only the fallback function.

Medium/Minor severity

None found.

Enhancements

Use solidity time literals instead of declared constants

This definition is redundant due to the supported time literals in solidity like **1 years**:

```
uint constant SECONDS_IN_YEAR = 31536000;
```

Use SafeMath functions like methods for unsigned integers

```
Since SafeMath is a library you may declare:

using SafeMath for uint;
```

And instead of writing this:

```
uint tokens = (SafeMath.mul(_value, 960));
```

You can write this:

```
uint tokens = _value.mul(960);
```

These calls can also be chained, improving overall clarity.

Conclusion

The contract was simple, consistent and straightforward, which is what we always recommend. It is always good to see code reuse. In particular, it is good to reuse already tested token implementations, like in this case.

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/etherparty-token-

EtherParty Smart Contract Security Audit

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

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

token-smart-contract-security-audit-coinfabrik/)

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

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

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

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

RCN BasicCosigner Smart Contract Security Audit

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

Coinfabrik security audit's team was asked to audit the BasicCosigner contract. This contract is part...

(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...

Tags:

bbl security audit (https://blog.coinfabrik.com/tag/bbl-security-audit/)

beluga pay security audit

SHARE ON

(https://www.facebook.com/sharer/sharer.php?u=https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/beluga-pay-bbi-security-audit/)

(https://twitter.com/intent/tweet?text=Beluga%20Pay%20(BBI)%20Security%20Audit&url=https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/beluga-pay-bbi-security-audit/)

(https://pinterest.com/pin/create/button/?url=&media=https://blog.coinfabrik.com/wp-

(https://blog.coinfabrik.com/tag/beluga-pay-security-audit/)

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

← PREVIOUS ARTICLE

Survey of Blockchain Storage and Computing Services (https://blog.coinfabrik.com/blockchain/blockchain-architecture/survey-blockchain-storage-computing-services/)

NEXT ARTICLE →

Rightmesh Token Sale Smart Contract Audit (Master) (https://blog.coinfabrik.com/smart-contracts/smart-contract-audit-smart-contracts/rightmesh-token-sale-smart-contract-audit-master/)

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://blog.coinfabrik.com/feed/)

(https://ar.linkedin.com/company/coinfabrik)

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

(https://twitter.com/coinfabrik)

(https://www.youtube.com/channel/UC2GmjCr7aEz-il31kqOy9aw)

(https://www.facebook.com/CoinFabrik/)

(https://www.reddit.com/r/CoinFabrik/)

(https://github.com/coinfabrik)

Made with ♥ in Buenos Aires, Argentina.