SECURITY AUDIT OF BCNX SMART CONTRACTS

PUBLIC REPORT

OCT 11, 2019

✔erichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology >> Forward

EXECUTIVE SUMMARY

This Security Audit Report prepared by Verichains Lab on Jul 19, 2019. We would like to thank ORITECH to trust Verichains Lab to audit smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the smart contracts. The scope of the audit is limited to the source code files provided to Verichains Lab on JUL 15, 2019. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

The assessment identified some issues in BCNX smart contracts code. Overall, the code reviewed is of good quality, written with the awareness of smart contract development best practices.

CONTENTS

Executive Summary 2
Acronyms and Abbreviations 4
Audit Overview 5
About ORITECH 5
Scope of the Audit 5
Audit methodology 6
Audit Results 7
Vulnerabilities Findings 8
MEDIUM Mixed implement of ERC223 and ERC20's variants 8
MEDIUM TokenVesting._vestedAmount logic's correctness depends on constant 8
LOW Misleading comment 8
Conclusion 10
Limitations 10
Appendix I 11

ACRONYMS AND ABBREVIATIONS

Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
ETH (Ether)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
EVM	Ethereum Virtual Machine.

AUDIT OVERVIEW

ABOUT ORITECH

ORITECH Technology Joint Stock Company, or ORITECH Corp. for short, was officially established in June 2008 with the enthusiasm and dynamism of a group of experts with many years of experience in the field of Telecommunications - IT.

From May 2014, ORITECH Technology Joint Stock Company officially changed its name to ORITECH Joint Stock Company, with the criterion of continuing to develop its strengths in the field of Telecommunications - IT and orienting technology solutions for telecommunications service operators as well as organizations and businesses.

The goal is to build ORITECH into an organization with good work capacity, able to acquire new technology techniques and transfer them to customers. ORITECH is and will always be willing to learn, serve and share with customers with all their knowledge and experience.

Solutions and services provided to customers will be ORITECH's pride. Any request is analyzed, proposed solutions and implemented in compliance with professional work processes to ensure that customer requirements are met at the best level.

ORITECH team works together in a modern and dynamic environment where new ideas are encouraged and respected. In the perspective of an organization where people will find their own place. The value is shared fairly in ORITECH's organization.

Constantly improving and ensuring commitments to customers are the factors that will make ORITECH become one of the best options for all needs in the field of Telecommunications - IT.

SCOPE OF THE AUDIT

This audit focused on identifying security flaws in code and the design of the smart contracts. It was conducted on file *Bcnx_flat.sol* sent to VeriChains on Jul 15, 2019.

Source File	SHA256 Hash
Bcnx_flat.sol	e491a12195671c8d74480aef010f322bdd324ad95b598fbf58920eaf9adb8ebb

ORITECH fixed all reported problems reported by VeriChains in version 4 of the contract, conducted on file Bcnx_flat_v4.docx sent to VeriChains on Jul 29, 2019.

AUDIT METHODOLOGY

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Integer Overflow and Underflow
- TimeStamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories, depending on their criticality:

LOW	An issue that does not have a significant impact, can be considered as less important
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.

AUDIT RESULTS

BcnxToken contract implements normal ERC223 standards with some other ERC20 standards features, almost features are reused from library codes with manual tuning (almost undocumented). The contract also implements vesting feature using TokenVesting contract, every calculations are handled by SafeMath library that also implicitly block invalid transfer operations. Library codes like SafeERC223 is copied and changed only in functions that BcnxToken used (safeTransfer) and leave other functions untouched.

In summary all the features are implemented carefully, we only identified some unclear logic and documenting problems in next section.

After receiving the original report on Jul 21, 2019, ORITECH kept communication with VeriChains and fixed all reported problems in contract version 4 on Jul 29, 2019.

VULNERABILITIES FINDINGS

MEDIUM
MIXED IMPLEMENT OF ERC223 AND ERC20'S VARIANTS

ERC223 token contract notifies the receiver whenever they receive token if the receiver is a contract, but its specification wrote nothing about other transfer events like mint. Actually, mint is an extended feature from ERC20's standards that also send token to a specified address from token pool (or from nowhere), it's better to implement `ERC223ReceivingContract` notification for **`_mint`** function.
In Bcnx contract, **`_mint`** is only called at constructor and its internal function so it cannot be called after the contract creation. The logic in this case is ok if that exception is intended (`msg.sender` and `vault` do not need to handle `ERC223ReceivingContract` feature). The contract ERC223 is documented as *Reference implementation of the ERC223 standard token*. So this may be a problem when someone copy this implement for their own.

ORITECH fixed this problem in v4 contract.

MEDIUM
TOKENVESTING._VESTEDAMOUNT LOGIC'S CORRECTNESS DEPENDS ON CONSTANT

Function **`_vestedAmount`** in **`TokenVesting`** contract was implemeneted in such a way that its correctness depends on `LOCK_END` and `AMOUNT_PER_RELEASE` constants, but the constraint did not documented anywhere in the contract. More specifically, the constraint:

$$(LOCK\_END.length - 1) * AMOUNT\_PER\_RELEASE \leq token.balanceOf(this) - \_released[token]$$

must always be hold or the beneficiary will be able to claim only after last lock period.

RECOMMENDED FIXES

There are 2 options:

- Document the constraint into **`TokenVesting`** contract, also reference the constraint in documenting of initial transfer of `TEAM_TOKEN`.
- Change the **`_vestedAmount`** function to return max available amount.

ORITECH fixed this problem in v4 contract.

LOW
MISLEADING COMMENT

Function **`_transfer`** of **ERC223** contract contains misleading comment, the function itself is ERC223's transfer function, not the backward fallback function.

```
function _transfer(address _from, address _to, uint256 _value, bytes memory _data) internal {
    // Standard function transfer similar to ERC20 transfer with no _data .
    // Added due to backwards compatibility reasons .
    uint codeLength;
```

RECOMMENDED FIXES

- Remove the comment.

ORITECH fixed this problem in v4 contract.

CONCLUSION

Bcnx smart contracts have been audited by Verichains Lab using various public and in-house analysis tools and intensively manual code review. The assessment identified some issues in Bcnx smart contracts code. Overall, the code reviewed is of good quality, written with the awareness of smart contract development best practices.

LIMITATIONS

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

APPENDIX I

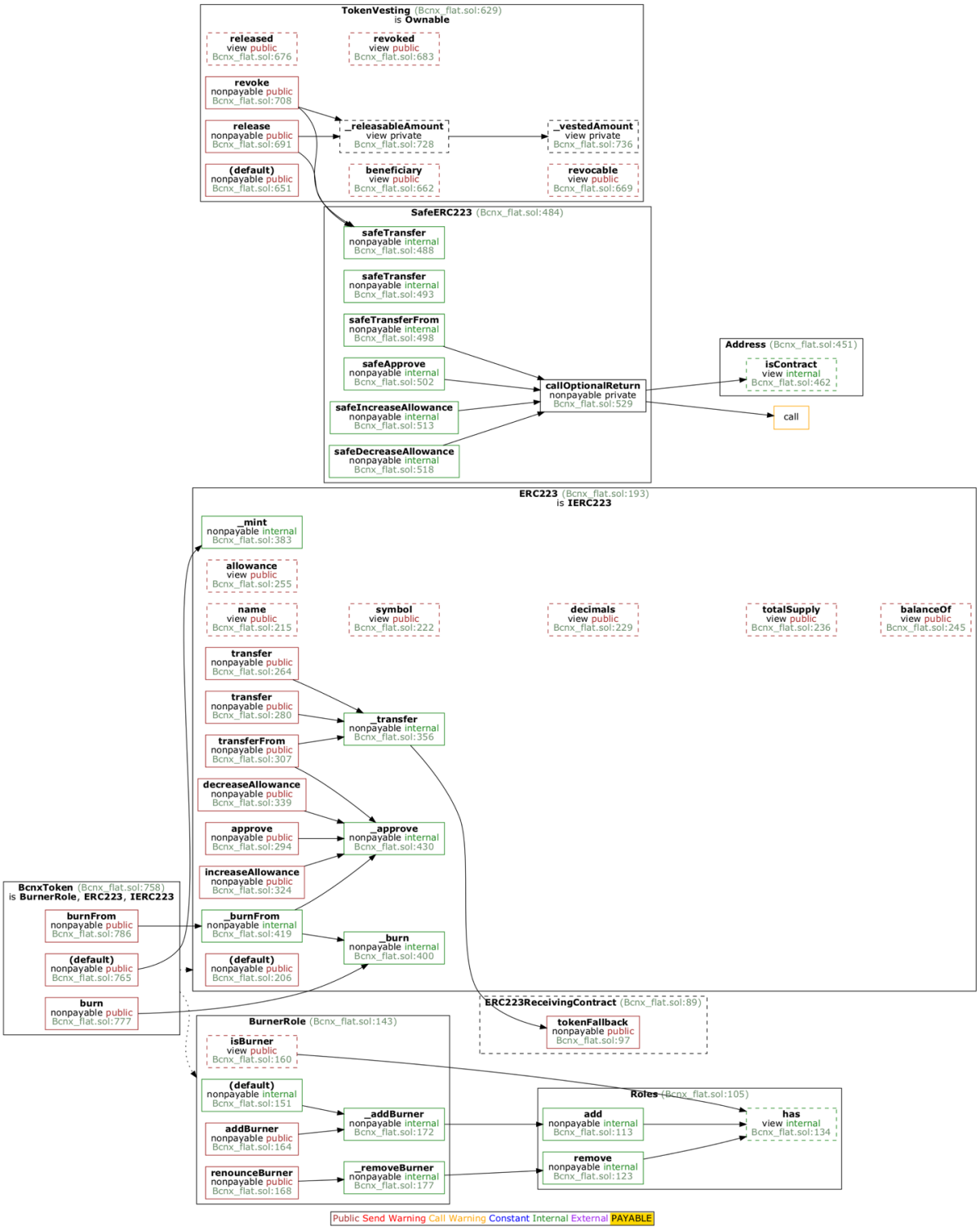

Figure 1 Contract call graph