

SMART CONTRACT

Security Audit Report

Project: Blockton Network
Website: <https://blocktoncoin.com>
Platform: Blockton Network
Language: Go
Date: January 25th, 2023

Contents

About EtherAuthority	4
Executive Summary	5
Review Summary	6
Manual Review Notes	7
• Introduction	7
• Documentation	7
• Areas of Concern	8
• Audit Summary	9
Severity Definitions	10
Audit Findings	11
Blockton Network Implementation Analysis	13
• Wallet Account Model	13
• Consensus	13
• Incentive Model	13
• Economic Model	13
Test Cases and Coverage	14
Conclusion	15
Our Methodology	16
Disclaimers	18
Appendix	19

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO THE PUBLIC AFTER ISSUES ARE RESOLVED.

About EtherAuthority

Founded in 2018 by blockchain experts and tech enthusiasts, EtherAuthority is a technology-led blockchain security company. Its mission is to prove the security and correctness of smart contracts and blockchain protocols through different approaches and detection methods, including manual, static, and dynamic analysis.

The official website is [EtherAuthority.io](https://etherauthority.io) and All the portfolio and public resources can be referred at: <https://github.com/etherauthority>

The company's team of seasoned engineers and security auditors apply testing methodologies and verifications to ensure projects are checked against known attacks and potential vulnerabilities. To date, EtherAuthority has provided high-quality auditing and consulting services to 400+ clients, including Catecoin, MainnetZ and HyperonChain.

The company customizes its engineering tool kits and applies cutting-edge research on smart contracts to each client's project to ensure a high quality delivery. As it continues to leverage technologies from blockchain and smart contracts, the EtherAuthority team will continue to support projects as a service provider and collaborator.

To verify the authenticity of this document, please refer to the official website, or github, or the social media announcements. Or, simply reach out to us: contact@etherauthority.io

Executive Summary

This report has been prepared for **Blockton Network** to review the implementation, security and soundness of their **Blockton Network system**. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Review the implementation and security of the **consensus** mechanism.
- Review the implementation and security of the **transaction** mechanism.
- Review the implementation and security of the **utxo** model.
- Review the **incentive** model.
- Review the **economic** model.
- Review the implementation of the **Clique based PoA** algorithm.

Review Summary

SECURITY LEVEL 	This report has been prepared as a product of the Code Audit request by Blockton Network. This audit was conducted to discover issues and vulnerabilities in the source code of Blockton Network.	
	TYPE	Smart Contract Platform
	SOURCE CODE	https://github.com/blocktoncoin/blockton-network/
	GITHUB COMMIT	7a10ec3f4dfce7c5f79faf8bbbc56a8f5a467b1e
	ALGORITHM	Clique based PoA
	PLATFORM	Blockton Network
	LANGUAGE	Go
	DELIVERY DATE	January 25th, 2023
	METHODS	Dynamic Analysis, Static Analysis, and Manual Review, has been performed.

Manual Review Notes

Introduction

The EtherAuthority team has been engaged by the Blockton team to audit the design and implementations of its system. The audited source code links:

<https://github.com/blocktoncoin/blockton-network/>

Specifically, we examined the Git revisions for our initial review:

7a10ec3f4dfce7c5f79faf8bbbc56a8f5a467b1e

The goal of this audit is to review Blockton implementation of its core mechanisms general design and architecture, study potential security vulnerabilities, and uncover bugs that could compromise the software in production.

Documentation

We used the following sources in respect to our work:

1. Website: <https://blocktoncoin.com>
2. Explorer: <https://blocktonscan.com>
3. Docs: <https://docs.blocktonscan.com>
4. Audit Scope : <https://github.com/blocktoncoin/blockton-network>

Areas of Concern

Our investigation focused on the following areas:

- Correctness of the protocol implementation;
- Attack vectors related to building, running & maintaining the nodes (eg version upgrades, downloading new clients, fork notifications);
- User funds are secure on the blockchain and cannot be transferred without user permission;
- Vulnerabilities within each component as well as secure interaction between the network components;
- Correctly passing requests to the network core;
- Data privacy, data leaking, and information integrity;
- Key management implementation: secure private key storage and proper management of encryption and signing keys;
- Handling large volumes of network traffic;
- Resistance to DDoS and similar attacks;
- Aligning incentives with the rest of the network;
- Vulnerabilities, potential misuse, and gaming of smart contracts;
- Any attack that impacts funds, such as draining or manipulating of funds;
- Mismanagement of funds via transactions;
- Inappropriate permissions and excess authority;
- Secure communication between the nodes;
- Special token issuance model; and
- Anything else as identified during the initial analysis phase.

Audit Summary

The results of the review and automated tools along with the manual examination of the code bases provided with a number of relevant findings regarding the application reviewed. The codebase in scope was mainly in **Go language**, as the project's chain, proof of stake algorithm and VM, and a small part in **Javascript and C programming language** regarding the layer and functionality.

Starting off with the **Go codebase** and the **blocktoncoin/blockton-network** repository, the audit has found the code base to be in a very high level of code design and implementation and findings are language related with no severity.

Moving forward to the Clique based PoA algorithm, the audit has examined the codebase on the consensus/clique/ folder. Alongside with the documentation provided by the team, the audit has checked the implementation manually and found it to be accurate to the specifications given.

The audit examined all other folders for design and implementation correctness and found it to be very well treated. Due to the complexity of the mechanism the audit was not able to summarize in depth about the security of the implementation as it was not possible to be addressed within the timeline of the scope. To conclude, the automated tools and manual review did not raise any issues.

To summarize, the audit has come to the conclusion that the coding team has done stellar work regarding the Go implementation of the chain, the proof of stake and VM, using the language best practices and implementing the designs at a very high level.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

General Comments

During our review, we found the Blockton code to be well-structured and clear to follow. Blockton demonstrates organized modular architecture, an effort to adhere to standardized development processes, and has adequate unit test coverage along with an open access test network environment - all of which are practices that are helpful for reducing the risk of security issues.

Blockton also maintains a considerable amount of documentation and appears to proactively engage with the Blockton community by maintaining an open source code base, which supports independent security reviews, in addition to public communications channels, including Telegram, which encourage community participation. Both of these efforts help to encourage the discoverability of security issues and complement independent reviews like this one.

However, Blockton's use of a Proof of Authority (PoA) consensus algorithm exposes it to a host of potential attacks that have not yet been identified, largely due to PoA being a fairly new approach to achieving blockchain consensus. This exposure is inherent given that design and development of resilient systems are still in the early stages and have yet to be extensively researched. Systems that do run PoA in production face criticism for being too centralized, not addressing the "nothing-at-stake problem", and various complexities with regard to block reorganization.

Blockton has taken reasonable steps to minimize the inherent risk with PoA, including soliciting an audit of their codebase and adding mechanisms like stacking and slashing. Precautions like stacking and slashing are incorporated to incentivize good behavior and discourage selfishness by network participants, however, it is not yet clear if these will be sufficient or stable throughout PoA network changes.

The use of PoA by Blockton and other comparable PoA algorithms in other blockchains will likely require close observation and potentially require adjustments to the mechanisms in place over time.

Finally, simplification of core components of the Blockton codebase by removing unused code should be a continuous effort. We acknowledge that such efforts are undertaken incrementally and iteratively, in order to increase code readability and allow for better review of code quality.

Critical Severity

No Critical severity vulnerabilities were found.

High Severity

No High severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No Low severity vulnerabilities were found.

Very Low / Informational / Best practices:

(1) Proof of Authority (PoA) consensus is used. This consensus algorithm increases the centralization. Although the Blockton team has used Polygon's BOR algorithm to increase the decentralization. But again, usage of PoA based consensus is not very popular.

Blockton Network Implementation Analysis

Wallet Account Model

We have reviewed the account model. We have examined the related codebase, the modules used and all related functionality and **we haven't found any discrepancies from the code.**

Consensus

The Blockton consensus algorithm is a variant of the Clique (PoA) Consensus. It aims to increase transaction processing throughput, decrease transaction confirmation latency, and enhance security by addressing the selfish-mining strategy.

All parts of the specification were found to be present and well translated to the code. We have found it achieves these goals without introducing any new security issues.

Incentive Model

The Blockton incentive model is derived from its consensus. It shares some properties with the Polygon's BOR Consensus - namely miners are incentivized to mine blocks by issuing block rewards. **We have not found any game-theoretical issues in the Blockton incentive model.**

Economic Model

The economic model is a subset of the incentive model of the system. The combination of the consensus model plus the Clique (PoA) algorithm creates a Blockton security environment around the economic model.

Since the consensus model addresses some key problems found in many blockchains, the addition of a custom BOR algorithm from Polygon **hardens the model to an elevated security level. We have not found any issues arising out of the economics of the system.**

Test Cases and Coverage

Considering the fact that the team has provided evidence of stellar performance on the code design and implementation, the small percentage of code coverage does not reflect the overall performance of the tests.

Also, the code is forked from Go-Ethereum 1.10.26 version, their test cases and coverages were considered automatically.

Conclusion

We were given a contract code in the form of a github link and we have used all possible tests based on given objects. We have not observed any major vulnerabilities in the code.

So it is good to go for the mainnet deployment.

Since possible test cases can be unlimited for such blockchain smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

The blockchain code within the scope was manually reviewed and analyzed with static analysis tools. These tools provided mostly false positive results and thus they can be safely ignored.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

The security state of the reviewed blockchain code, based on standard audit procedure scope, is **“Secured”**.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can't guarantee explicit security of the audited smart contracts.

Appendix

We compared the entire source code of blockton network with go-ethereum source version 1.10.26. And we inspected differences manually for each change. Then we compiled with Go version 1.17 and tested by running geth, which worked successfully in all aspects. Please find the screenshot below for the same.

```
[INFO [01-25|03:03:03.030] - Total terminal difficulty: 5875000000000000000000000
[INFO [01-25|03:03:03.030] - Merge netsplit block: <nll>
-----
[INFO [01-25|03:03:03.030] Disk storage enabled for ethash caches dir=/home/shahkchandan/blockton-network-master/cmd/geth/.datafolder/geth/ethash count=3
[INFO [01-25|03:03:03.031] Disk storage enabled for ethash DAGS dir=/home/shahkchandan/.ethash count=2
[INFO [01-25|03:03:03.031] Initializing Ethereum protocol network=508 dsnr=ipnba
[INFO [01-25|03:03:03.066] Loaded most recent local header number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=53y10m2d
[INFO [01-25|03:03:03.067] Loaded most recent local full block number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=53y10m2d
[INFO [01-25|03:03:03.067] Loaded most recent local fast block number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=53y10m2d
[INFO [01-25|03:03:03.067] Loaded local transaction journal transactions=0 dropped=0
[INFO [01-25|03:03:03.067] Regenerated local transaction journal transactions=0 accounts=0
[INFO [01-25|03:03:03.067] Chain post-merge, sync via beacon client
[INFO [01-25|03:03:03.067] Gasprice oracle is ignoring threshold set threshold=2
[MARN [01-25|03:03:03.067] Engine API enabled protocol=eth
[INFO [01-25|03:03:03.067] Starting peer-to-peer node instance=Geth/v1.10.26-stable/Linux-amd64/go1.17.13
[INFO [01-25|03:03:03.075] New local node record seq=1,674,644,531,490 id=22e4d258a8a77ffe ip=127.0.0.1 udp=30303 tcp=30303
[INFO [01-25|03:03:03.076] IPC endpoint opened url=/datafolder/geth.ipc
[INFO [01-25|03:03:03.076] Loaded JWT secret file path=/home/shahkchandan/blockton-network-master/cmd/geth/.datafolder/geth/jwtsecret crc32=0xb27a9a98
[INFO [01-25|03:03:03.077] Websocket enabled urlws://127.0.0.1:8551
[INFO [01-25|03:03:03.077] HTTP server started endpoints=127.0.0.1:8551 auth=true prefix=cors=localhost whosts=localhost
-----
[MARN [01-25|03:03:03.077] Referring to accounts by order in the keystore folder is dangerous!
[MARN [01-25|03:03:03.077] This functionality is deprecated and will be removed in the future!
[MARN [01-25|03:03:03.077] Please use explicit addresses! (can search via `geth account list`)
-----
[MARN [01-25|03:03:03.077] -----
Unlocking account 0 / Attempt 1/3
[INFO [01-25|03:03:03.081] Started P2P networking self=enode://c7f0670af51183191dbf7cd066822cb7f3bb160bc16409f23aff188bc05e7ae3ca459a926b34de0481d46a7a39872535C867adbf77f72480dfbe9c5bd549c3d4287507127.0.0.1:30303
[INFO [01-25|03:03:05.438] Unlocked account address=0x86f091b741c7b7B20291888Ea7c850886716Be5
[INFO [01-25|03:03:05.483] Etherbase automatically configured address=0x86f091b741c7b7B20291888Ea7c850886716Be5
Welcome to the geth JavaScript console!

instance: Geth/v1.10.26-stable/Linux-amd64/go1.17.13
coinbase: 0x86f091b741c7b7B20291888Ea7c850886716Be5
blocks: 0 (Wed Dec 31 1999 16:00:00 GMT+0800 (PST))
datadir: /home/shahkchandan/blockton-network-master/cmd/geth/.datafolder
modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> INFO [01-25|03:03:07.459] New local node record seq=1,674,644,531,491 id=22e4d258a8a77ffe ip=59.153.97.87 udp=64788 tcp=30303
[INFO [01-25|03:03:13.303] Looking for peers peercount=0 tried=111 static=0
```

