#Hashlock.

Bethel

(Token)

SMART CONTRACT

Security Audit

Performed on Contracts:

BETHEL.sol

MD5 Hash:243656a4cf85b1933abd7c3797016eeb

Platform

Polygon

hashlock.com.au

JUNE 2024

Table of Contents

CAUTION

THIS DOCUMENT IS A SECURITY AUDIT REPORT AND MAY CONTAIN CONFIDENTIAL INFORMATION. THIS INCLUDES IDENTIFIED VULNERABILITIES AND MALICIOUS CODE THAT COULD BE USED TO COMPROMISE THE PROJECT. THIS DOCUMENT SHOULD ONLY BE FOR INTERNAL USE UNTIL ISSUES ARE RESOLVED. ONCE VULNERABILITIES ARE REMEDIATED, THIS REPORT CAN BE MADE PUBLIC. THE CONTENT OF THIS REPORT IS OWNED BY HASHLOCK PTY LTD FOR THE USE OF THE CLIENT.

Executive Summary

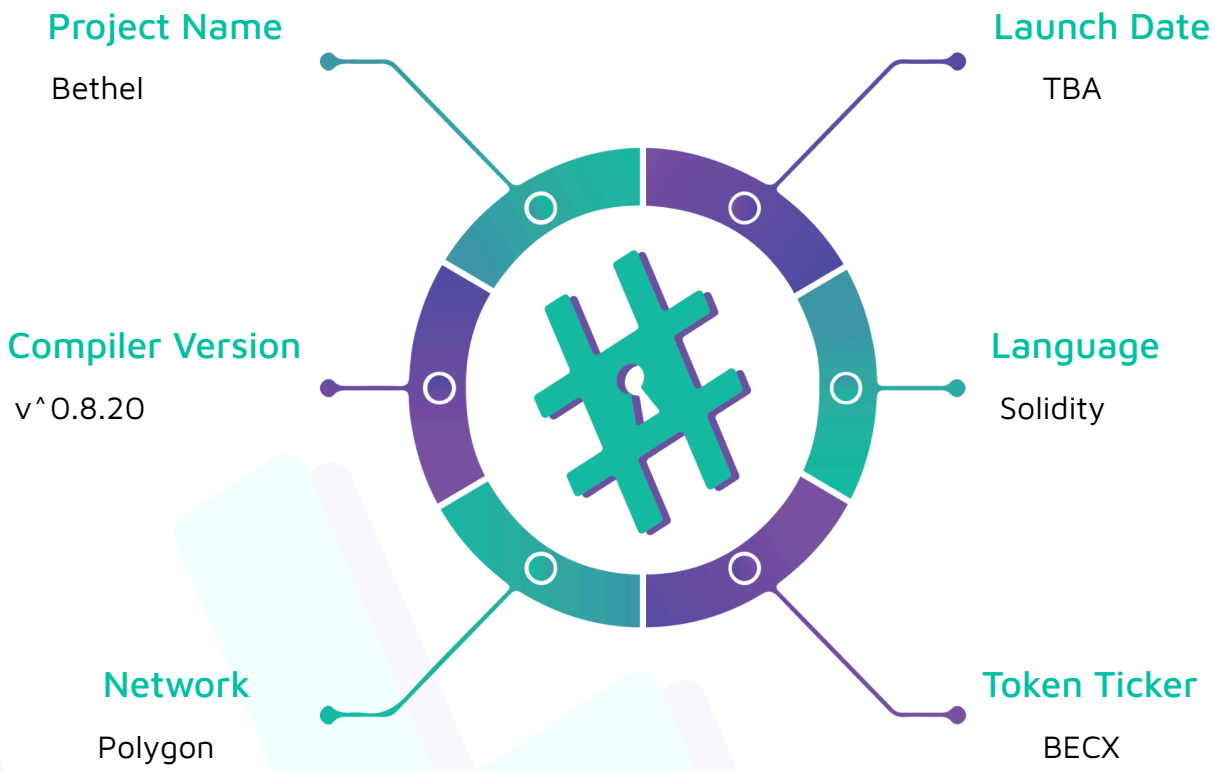
The Bethel team partnered with Hashlock to conduct a security audit of their ERC20 smart contract for BECX token. Hashlock manually and proactively reviewed the code to ensure the project's team and community that the deployed contracts were secure.

Project Context

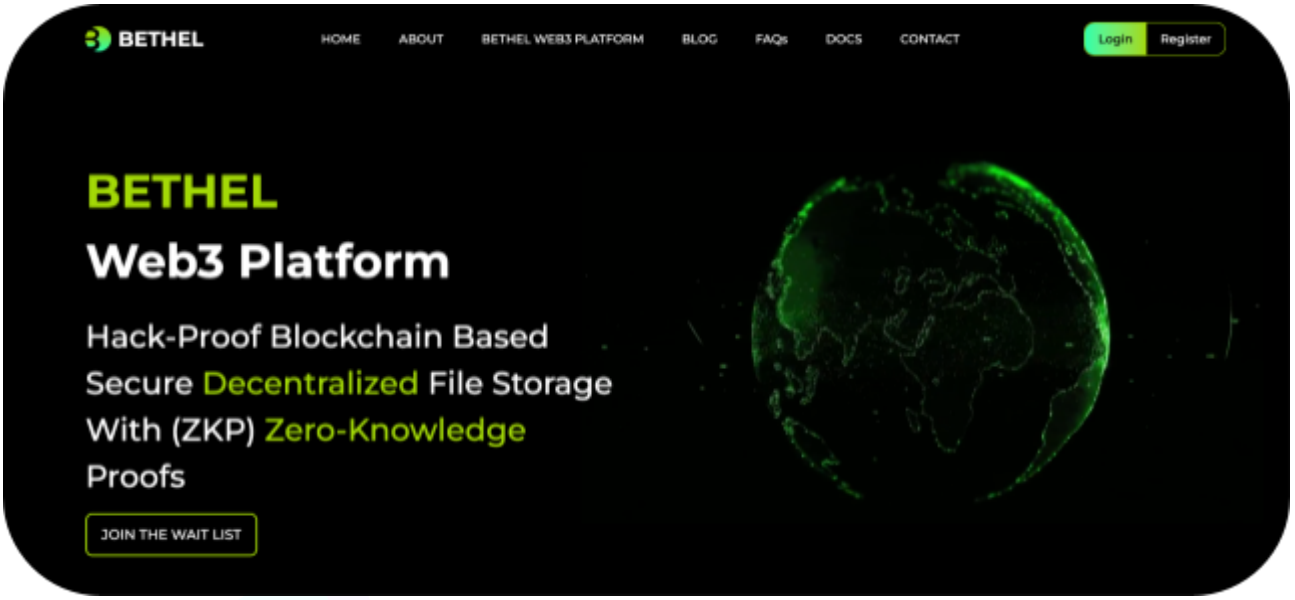
BECX is the utility token of the Bethel platform through which all the activities of this platform are carried out.

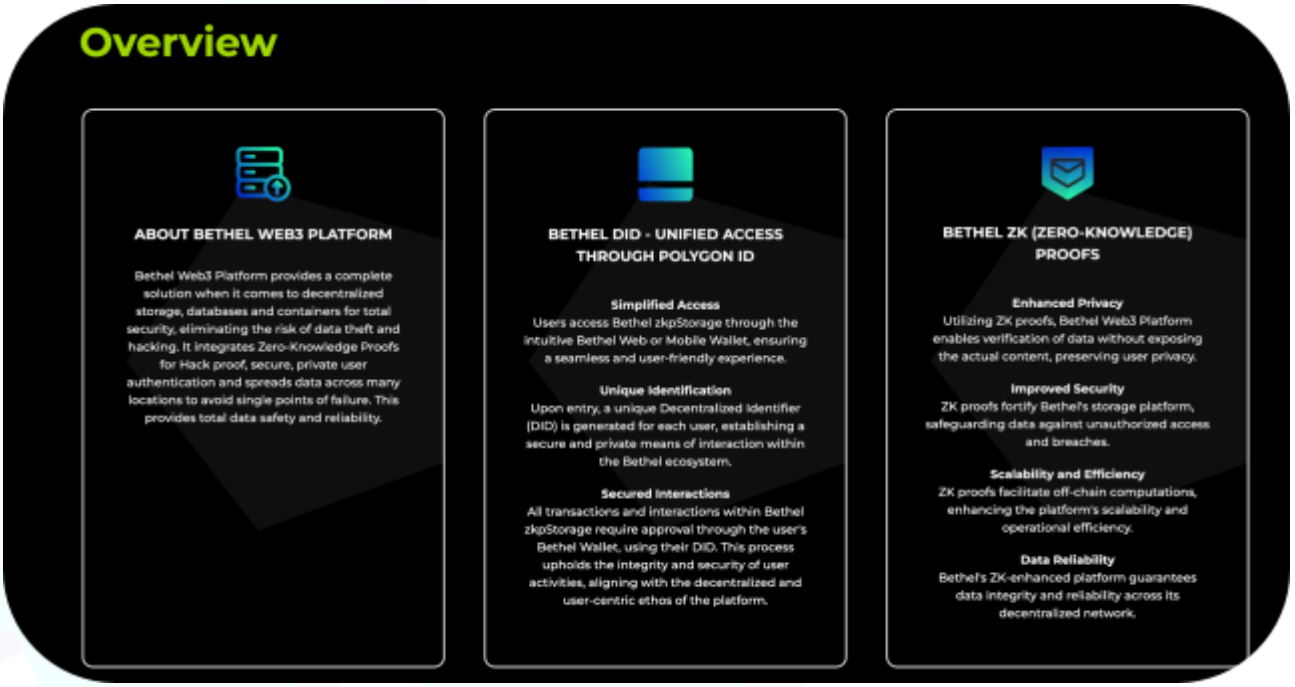
Project Name: Bethel
Compiler Version: ^0.8.20
Website: https://bethelnet.io/
Logo:

Visualised Context:



Project Visuals:





Audit scope

We at Hashlock audited the solidity code within the Bethel token, the scope of work included a comprehensive review of the smart contracts listed below. We tested the smart contracts to check for their security and efficiency. These tests were undertaken primarily through manual line-by-line analysis and were supported by software-assisted testing.

Description	**Bethel Protocol Smart Contracts**
Platform	**Polygon / Solidity**
Audit Date	**June, 2024**
Contract 1	Bethel.sol
Contract 1 MD5 Hash	243656a4cf85b1933abd7c3797016eeb

Security Rating

After Hashlock's Audit, we found the smart contracts to be **"Secure"**. The contracts all follow simple logic, with correct and detailed ordering. They use a series of interfaces, and the protocol uses a list of Open Zeppelin contracts. We have identified some vulnerabilities that need to be addressed prior to launch.

The 'Hashlocked' rating is reserved for projects that ensure ongoing security via bug bounty programs or on-chain monitoring technology.

All issues uncovered during automated and manual analysis were meticulously reviewed and applicable vulnerabilities are presented in the Audit Findings section.

All vulnerabilities we have identified have been resolved or acknowledged.

Hashlock found:

0 High-severity vulnerabilities

1 Medium-severity vulnerabilities

0 Low-severity vulnerabilities

0 Gas Optimisations

Caution: *Hashlock's audits do not guarantee a project's success or ethics, and are not liable or responsible for security. Always conduct independent research about any project before interacting.*

Intended Smart Contract Behaviours

Claimed Behaviour	Actual Behaviour
Bethel.sol - Low complexity ERC20 Token.	**Contract achieves this functionality.**

Code Quality

This Audit scope involves the smart contracts of the Bethel token, as outlined in the Audit Scope section. All contracts, libraries, and interfaces mostly follow standard best practices to help avoid unnecessary complexity that increases the likelihood of exploitation.

The code is very well commented on and closely follows best practice nat-spec styling. All comments are correctly aligned with code functionality.

Audit Resources

We were given the Bethel token smart contract code in the form of zip files.

As mentioned above, code parts are well-commented. The logic is straightforward, and therefore it is easy to quickly comprehend the programming flow and the complex code logic. The comments help us understand the overall architecture of the protocol.

Dependencies

Per our observation, the libraries used in this smart contracts infrastructure are based on well-known industry-standard open-source projects.
Apart from libraries, its functions are used in external smart contract calls.

Severity Definitions

Significance	Description
High	High-severity vulnerabilities can result in loss of funds, asset loss, access denial, and other critical issues that will result in the direct loss of funds and control by the owners and community.
Medium	Medium-level difficulties should be solved before deployment, but won't result in loss of funds.
Low	Low-level vulnerabilities are areas that lack best practices that may cause small complications in the future.
Gas	Gas Optimisations, issues, and inefficiencies

Audit Findings

Medium

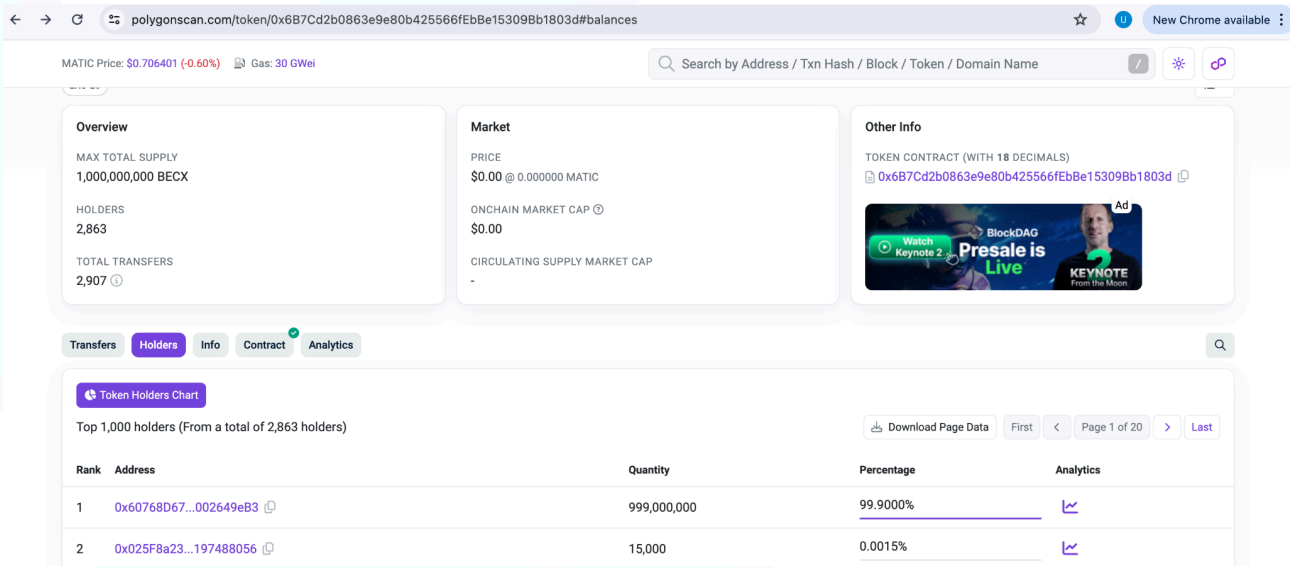
[M-01] Centralization Risk

Description

A centralization risk is present as 99% of the minted tokens are held by a single account (the deployer of the contract). This creates a potential single point of failure. A malicious or a compromised owner of the tokens can execute attacks to manipulate the token price.

Vulnerability Details

In the BECX token's Polygonscan page, it can be verified that the address 0x60768D677fB19d6Ee8c505ABfeC5b7F002649eB3 holds more than 99% of the minted tokens.



Impact

The centralization of tokens can lead to trust issues among investors and users, as well as potential regulatory concerns. It may also affect the market liquidity and stability of the token.

Recommendation

It is recommended to implement a more distributed token distribution strategy to mitigate this risk. This could include a vesting schedule, distributing tokens to multiple wallets, or implementing a decentralised governance mechanism.

Note:

During the audit, it was observed that the vesting schedule had not yet commenced at the beginning of the audit period. However, the vesting schedule was initiated during the auditing process, which allowed for its review and confirmation of proper implementation within the audit scope.

Status

Resolved

Centralisation

The project values security and utility over decentralisation, but still inherits aspects of decentralisation.

There are no owner executable functions to mint extra tokens or backdoor functions to control token liquidity. This shows that the Bethel token's supply cannot be controlled by admins after its launch.

Conclusion

After Hashlock's analysis, the Bethel token (BECX) appears to have a sound and well-tested code base, now that our vulnerability findings have been resolved and acknowledged. Most of the code is correctly ordered and follows industry best practices, with well-commented code to aid in understanding the protocol's architecture.

It is important to note that the vesting schedule for the Bethel token (BECX) had not yet commenced at the time the audit began. However, the vesting schedule was initiated during the auditing period, which allowed us to observe its implementation and initial operations. This timing ensured that the vesting process was thoroughly reviewed within the audit scope, confirming its adherence to the intended smart contract behaviors and security protocols.

Overall, to the best of our ability, Hashlock has not identified any further vulnerabilities. The Bethel token (BECX) smart contracts are deemed secure and ready for deployment, following the resolution of all identified issues.

Our Methodology

Hashlock strives to maintain a transparent working process and to make our audits a collaborative effort. The objective of our security audits are to improve the quality of systems and upcoming projects we review and to aim for sufficient remediation to help protect users and project leaders. Below is the methodology we use in our security audit process.

Manual Code Review:

In manually analysing all of the code, we seek to find 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 methodologies include manual code analysis, user interface interaction, and white box penetration testing. We consider the project's website, specifications, and whitepaper (if available) to attain a high-level understanding of what functionality the smart contract under review contains. We then communicate with the developers and founders to gain insight into their vision for the project. We install and deploy 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 undergo a robust, transparent process for analysing potential security vulnerabilities and seeing them through to successful remediation. When a potential issue is discovered, we immediately create an issue entry for it in this document, even though we still need to verify the feasibility and impact of the issue. This process is vast because we document our suspicions early even if they are later shown not to represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, and 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 analyse 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 scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the contracts details are made public.

Disclaimers

Hashlock's Disclaimer

Hashlock's team has analysed these smart contracts in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in the 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 is unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free 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.

Hashlock is not responsible for the safety of any funds and is not in any way liable for the security of the project.

Technical Disclaimer

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

About Hashlock

Hashlock is an Australian-based company aiming to help facilitate the successful widespread adoption of distributed ledger technology. Our key services all have a focus on security, as well as projects that focus on streamlined adoption in the business sector.

Hashlock is excited to continue to grow its partnerships with developers and other web3-oriented companies to collaborate on secure innovation, helping businesses and decentralised entities alike.

Website: hashlock.com.au
Contact: info@hashlock.com.au


#Hashlock.