Bitcoin EDenRich - audit #2

Preliminary Comments

CertiK Assessed on Jun 18th, 2025

CertiK Assessed on Jun 18th, 2025

Bitcoin EDenRich - audit #2

These preliminary comments were prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
ERC-20	Polygon (MATIC)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 06/18/2025	N/A

CODEBASE

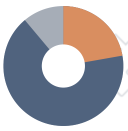
https://polygonscan.com/address/0x430f5d89ada20dd2b31fbde0232cf802c7d138c0

View All in Codebase Page

Highlighted Centralization Risks

- Privileged role can remove users' tokens
- Transfers can be paused
- Privileged role can mint tokens
- Initial owner token share is 100%
- Has blacklist/whitelist

Vulnerability Summary



9	0	0	0	0	9
Total Findings	Resolved	Partially Resolved	Acknowledged	Declined	Pending

🟧 2	Centralization	2 Pending	Centralization findings highlight privileged roles & functions and their capabilities, or instances where the project takes custody of users' assets.
🟥 0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧 0	Major		Major risks may include logical errors that, under specific circumstances, could result in fund losses or loss of project control.
🟨 0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

🟨	0	Minor		Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	6	Informational	6 Pending	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.
⬜	1	Discussion	1 Pending	The impact of the issue is yet to be determined, hence requires further clarifications from the project team.

TABLE OF CONTENTS | BITCOIN EDENRICH - AUDIT #2

CODEBASE | BITCOIN EDENRICH - AUDIT #2

Repository

https://polygonscan.com/address/0x430f5d89ada20dd2b31fbde0232cf802c7d138c0

AUDIT SCOPE | BITCOIN EDENRICH - AUDIT #2

1 file audited ● 1 file with Pending findings

ID	Repo	Commit	File	SHA256 Checksum
● BED	mainnet	0x430f5	BEDR.sol	b34cc48a6169bf3b94cd006b3ad00dab3811a0cd0725e7cb928f5f435f4dd4f0

APPROACH & METHODS | BITCOIN EDENRICH - AUDIT #2

This report has been prepared for Bitcoin to discover issues and vulnerabilities in the source code of the Bitcoin EDenRich - audit #2 project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | BITCOIN EDENRICH - AUDIT #2

Overview

The **Bitcoin EdenRich (BEDR)** project is an ERC20 token project. The focus of this audit is the token contract.

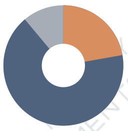
Privileged Functions

In the **Bitcoin EdenRich (BEDR)** project, the privileged roles are adopted to ensure the dynamic runtime updates of the project, which are specified in the `Centralization` finding.

The advantage of those privileged roles in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worth noting the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private keys of the privileged accounts are compromised, it could lead to devastating consequences for the project.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

FINDINGS | BITCOIN EDENRICH - AUDIT #2



9	0	2	0	0	0	6	1
Total Findings	Critical	Centralization	Major	Medium	Minor	Informational	Discussion

This report has been prepared to discover issues and vulnerabilities for Bitcoin EDenRich - audit #2. Through this audit, we have uncovered 9 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
BEA-02	**Initial Token Distribution**	**Centralization**	**Centralization**	● **Pending**
BEA-08	**Centralization Risks In BEDR.Sol**	**Centralization**	**Centralization**	● **Pending**
BEA-05	Missing Bounds Check In `lockState()`	Design Issue	Informational	● Pending
BEA-06	Potential Overflow In `lockAfter()` Function	Incorrect Calculation	Informational	● Pending
BEA-09	Local Variable Shadowing	Coding Style	Informational	● Pending
BEA-10	Outdated Versions Of Solidity	Language Version	Informational	● Pending
BEA-11	Potential Meaningless Argument For `pause/unpause`	Coding Issue	Informational	● Pending
BEA-12	Misleading Error Message	Coding Issue	Informational	● Pending
BEA-01	Discussion On Missing Frozen Address Check On Recipient	Design Issue	Discussion	● Pending

BEA-02 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization	● **Centralization**	**BEDR.sol: 104**	● **Pending**

Description

All of the `BEDR` tokens are sent to the contract deployer or one or several externally-owned account (EOA) addresses. This is a centralization risk because the deployer or the owner(s) of the EOAs can distribute tokens without obtaining the consensus of the community. Any compromise to these addresses may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution plan should be published in a public location that the community can access. The team should make efforts to restrict access to the private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

If the team could provide the initial token distribution information such as the link to the token distribution plan, multi-sig wallet, and signer addresses, the information would be verified and updated in the report.

BEA-08 | CENTRALIZATION RISKS IN BEDR.SOL

Category	Severity	Location	Status
Centralization	● Centralization	BEDR.sol: 114, 118, 135, 139, 150, 154, 177, 183, 226, 234, 243, 256, 266	● Pending

Description

In the contract `BEDR` , the role `owner` has authority over the functions below:

- `transferOwnership()` : Transfers ownership of the contract to a new account.
- `renounceOwnership()` : Renounces ownership.
- `pause()` : Pause all token transfers across the contract.
- `unpause()` : Unpause and allows transfers again.
- `freeze()` : Freezes the specified address, preventing it from sending tokens.
- `unfreeze()` : Unfreezes the address to restore its ability to transfer tokens.
- `mint()` : Mint arbitrary tokens to specific accounts.
- `burn()` : Burns tokens from a holder.
- `lock()` : Locks a `holder` `amount` tokens until an absolute `releaseTime`
- `lockAfter()` : Locks a `holder` `amount` tokens for a relative period of `afterTime`
- `unlock()` : Unlocks a specific locked `idx` for a `holder` before its scheduled release.
- `transferWithLock()` : Transfers tokens from the `owner` to a recipient and locks them until `releaseTime` .
- `transferWithLockAfter()` : Transfers tokens from the `owner` to a recipient and locks them for `afterTime` from current time.

Any compromise to the role `owner` may allow the hacker to take advantage of this authority and has the ability to pause the market, freeze or unfreeze any users, mint new tokens to chosen accounts, lock or unlock users' assets arbitrarily, and transfer any amount of tokens to addresses under their control.

Especially, the role `owner` has the authority to mint the token to arbitrary account. If a hacker gains access to the private key of the owner's wallet and uses the `mint` function to create a large number of tokens for their own address, they can then sell all of these tokens. This action can lead to a significant drop in the token price.

Additionally, any compromise to the `owner` account can give a hacker the ability to exploit this authority and manipulate accounts' balances. The attacker can use the `burn` function to burn the balances of any address or the `lock` function to freeze the balances of any addresses. This may pose a risk to the market price of the token.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

BEA-05 | MISSING BOUNDS CHECK IN `lockState()`

Category	Severity	Location	Status
Design Issue	● Informational	BEDR.sol: 221~223	● Pending

Description

The lockState function directly accesses `lockInfo[holder][idx]` without verifying whether the provided `idx` is within the bounds of the array.

Recommendation

Add an explicit bounds check to ensure that the index is within the range.

BEA-06 | POTENTIAL OVERFLOW IN `lockAfter()` FUNCTION

Category	Severity	Location	Status
Incorrect Calculation	● Informational	BEDR.sol: 238	● Pending

Description

In Solidity 0.5.8, arithmetic operations do not include automatic overflow checks. If `afterTime` is extremely large, the sum can overflow and wrap around to a low value. This would result in an prematurely expired lock.

```
uint256 rel=now+afterTime;
```

However, the function is protected by `onlyOwner` , meaning this risk is limited to misuse or misconfiguration by the contract owner.

Recommendation

Use `SafeMath.add()` for secure arithmetic.

```
uint256 rel = now.add(afterTime);
```

BEA-09 | LOCAL VARIABLE SHADOWING

Category	Severity	Location	Status
Coding Style	● Informational	BEDR.sol: 234, 266, 279	● Pending

Description

A local variable is shadowing another component defined elsewhere. This means that when the contract accesses the variable by its name, it will use the one defined locally, not the one defined in the other place. The use of the variable may lead to unexpected results and unintended behavior.

```
266      function transferWithLockAfter(address to,uint256 value,uint256 afterTime)
public onlyOwner returns (bool){
```

- Local variable `afterTime` in `BEDR.transferWithLockAfter` shadows the function `afterTime` in `BEDR`.

```
234      function lockAfter(address holder,uint256 amount,uint256 afterTime) public
 onlyOwner {
```

- Local variable `afterTime` in `BEDR.lockAfter` shadows the function `afterTime` in `BEDR`.

Recommendation

It is recommended to remove or rename the local variable that shadows another definition to prevent potential issues and maintain the expected behavior of the smart contract.

BEA-10 | OUTDATED VERSIONS OF SOLIDITY

Category	Severity	Location	Status
Language Version	● Informational	BEDR.sol: 5	● Pending

Description

Solidity frequently releases new compiler versions with improved security features and bug fixes. Using an outdated version prevents access to these enhancements and may leave the smart contract vulnerable to known issues.

Recommendation

It is recommended to deploy with Solidity version ^0.8.0, which offers benefits such as new language features, fewer bugs, and more efficient gas usage, ultimately enhancing code readability and maintainability. Additionally, use a simple pragma version that allows any of these versions. Consider using the latest version of Solidity for testing.

Reference: https://github.com/ethereum/solidity/releases.

BEA-11 | POTENTIAL MEANINGLESS ARGUMENT FOR `pause/unpause`

Category	Severity	Location	Status
Coding Issue	● Informational	BEDR.sol: 135	● Pending

Description

The emitted value argument `uint256 value` might be unnecessary and waste gas, cluttering the event log.

```
    function pause(uint256 value) public onlyOwner whenNotPaused {
        paused=true;
        emit Pause(value);                                          // emit 존재(BEA-05)
    }
    function unpause(uint256 value) public onlyOwner whenPaused {
        paused=false;
        emit Unpause(value);                                        // emit 존재(BEA-05)
    }
```

Recommendation

Recommend that the team double-check if this is the intended design or remove any unnecessary variables.

BEA-12 | MISLEADING ERROR MESSAGE

Category	Severity	Location	Status
Coding Issue	● Informational	BEDR.sol: 122	● Pending

Description

The `_transferOwnership()` reverts with "Already owner" when `newOwner == address(0)` – misleading for user and for off-chain scripts that parse revert reasons.

```
function _transferOwnership(address newOwner) internal {
    require(newOwner!=address(0),"Already owner");
    emit OwnershipTransferred(owner,newOwner);
    owner = newOwner;
}
```

Recommendation

Recommend correcting the error message.

BEA-01 | DISCUSSION ON MISSING FROZEN ADDRESS CHECK ON RECIPIENT

Category	Severity	Location	Status
Design Issue	● Discussion	BEDR.sol: 161~164	● Pending

Description

Both `transfer()` and `transferFrom()` fail to validate whether the recipient `to` address is frozen before performing the transfer.

As a result, tokens can be transferred into frozen accounts. We would like to know if this is intended design.

APPENDIX | BITCOIN EDENRICH - AUDIT #2

Finding Categories

Categories	Description
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Language Version	Language Version findings indicate that the code uses certain compiler versions or language features with known security issues.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Elevating Your Entire Web3 Journey

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we’re able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.