CHAIN TROOPERS

Energy Web Foundation

Block Reward of EWC system contracts

Security Assessment Report

July 10th, 2025

Version 1.1

CONFIDENTIAL

Table of Contents

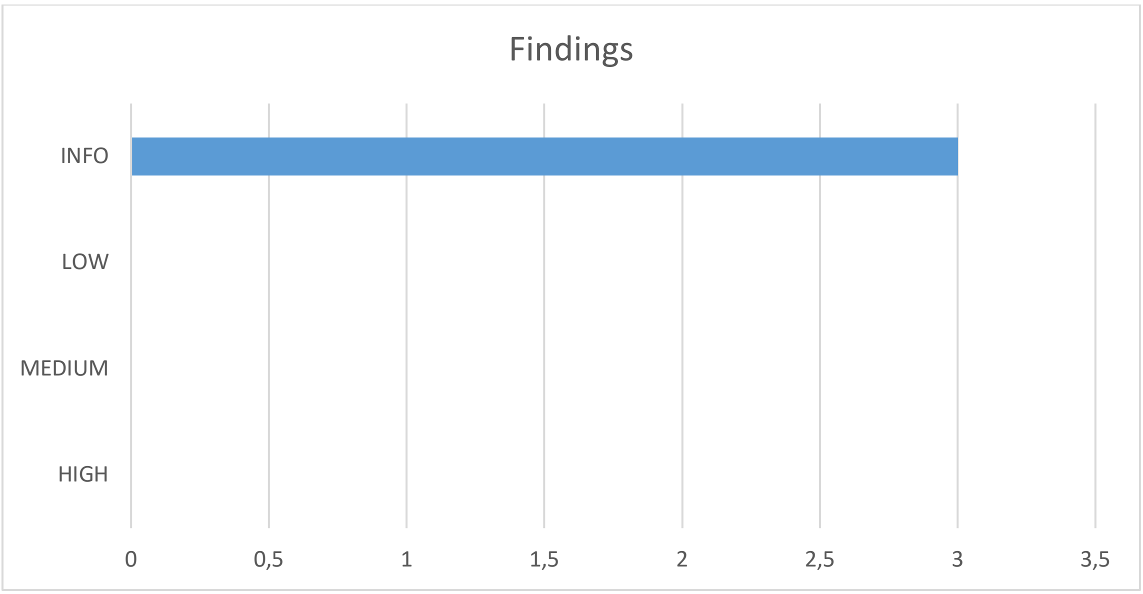
1 Executive Summary

1.1 Introduction

The report contains the results of Energy Web security assessment that took place from June 4th, 2025, to June 5th, 2025.

The in-scope component was the Block Reward system of the infrastructure contracts developed for EnergyWeb Chain (EWC) live launch. The security engineers performed an in-depth manual analysis of a specific change in the provided functionalities, and uncovered issues that could affect the confidentiality, the integrity, and the availability of the in-scope components.

Overall, the team identified three (3) informational issues that present no risk.



All the identified findings are presented in the report, including their impact and the proposed mitigation strategy, and are ordered by their severity. A retesting phase was carried out on July 10th, 2025, and the results are presented in Section 6.

1.2 Assessment Results

The in-scope changes modify the Block Reward contract so that it no longer mints any new tokens to *block authors (validators)* or to the *community fund*. All other logic (e.g. input validation, security controls, accounting counters) is left intact. After this change, transaction fees become the sole source of validators income on EWC (barring any off-chain arrangements).

The assessment results revealed that the modified in-scope application components were only affected by three (3) INFORMATIONAL issues bearing no security risk. The primary security considerations pertain to the deployment process (specifically, ensuring that all nodes upgrade simultaneously to avoid a consensus divergence), while the underlying codebase itself exhibits no inherent security weaknesses.

Specifically, the team determined that the "*BlockReward*" implementation was designed as an immutable, non-upgradeable contract and any change to the block-reward logic can only be implemented via a protocol-level (hard-fork) transition *("5.1.1 - Lack of a dedicated upgrade mechanism*"), which requires updating the spec to point to a new contract at a certain block. Consequently, any miscommunication or delay in adopting the updated chain spec could result in some nodes continuing to call the old "*BlockReward*" contract while others would have switched to the new one. This could cause inconsistent block-reward distributions, validator confusion, and, in extreme cases, a temporary chain split if blocks' validity rules start to diverge.

Additionally, it was found that the test suite for the "*ewc-system-contracts*" was developed under the assumption of the original S-curve and constant payout mechanisms; consequently, any test asserting a nonzero reward would no longer pass (*"5.1.2 - Tests were not updated*"). Failing tests could yield an immediate interruption to CI/CD workflows, as automated pipelines relying on the current test suite would report failures, preventing merges until tests are updated.

Finally, it was identified that the implementation still inherits from the "*SCurveProvider"* library and retains state variables (e.g., "*communityFundAmount*") that are no longer utilized in any reward calculations *("5.1.3 - Dead code was not removed").* The leftover code may cause confusion for future maintainers and incur slightly higher gas costs during the first deployment (e.g., CREATE operations).

In summary, this pull request successfully eliminates block rewards without introducing new security vulnerabilities. It reflects a policy change in EWC's token economics, effectively capping the token supply earlier than initially planned.

Chaintroopers recommend conducting this change with proper governance approval and a carefully orchestrated network upgrade. It is also advisable to address the INFORMATIONAL findings (clarifying the upgrade path, updating the test suite, and removing dead code) to enhance the system's maintainability and operational clarity.

1.2.1 Retesting Results

Results from retesting carried out on July 2025, determined that two (2 out of 3) reported issues bearing no risk (INFORMATIONAL) were sufficiently addressed (see sections *5.1.1* and *5.1.2*).

The rest one (1 out of 3 total findings) INFORMATIONAL finding was marked as risk-accepted (see section *5.1.3*). More information can be found in Section 6.

1.3 Summary of Findings

The following findings were identified in the examined source code:

Vulnerability Name	Status	Retest Status	Page
Lack of a dedicated upgrade mechanism	INFO	CLOSED	13
Tests were not updated	INFO	CLOSED	15
Dead code was not removed	INFO	Risk Accepted	17

2 Assessment Description

2.1 Target Description

The target of this assessment is a change in the Block Reward system of the infrastructure contracts developed for EnergyWeb Chain (EWC) live launch.

The Energy Web Chain (EWC) launched with a built-in Block Reward system contract that mints new EWT tokens at each block. This contract implements Parity's BlockReward interface and defines rewards for two (2) entities:

- **The Block Authors (Validators):** For each block, the validator who seals it receives a reward. The total issuance for validators was set to ~10 million EWT over the first 10 years, distributed on a discrete S-curve that starts higher and tapers over time. This was designed to incentivize and compensate validators for securing the network, alongside transaction fees.
- **The Community Fund:** A fixed portion of each block's reward is sent to a community fund (a multi-signature wallet initially controlled by EWF and later by the community). The community fund's purpose is to finance ecosystem development, grants, and other community-driven initiatives.

The block reward mechanism is embedded in EWC's protocol and consensus rules. The consensus engine (a Proof-of-Authority variant) automatically invokes the "*BlockReward*" contract each time a block is sealed, minting the new tokens and crediting the block author and community fund according to the contract logic.

The in-scope pull request "*Disable rewards*" modifies the Block Reward contract so that it no longer mints any new tokens to *block authors* or the *community fund.* All other logic (e.g. input validation, accounting counters) is left intact. This change halts the reward issuance prematurely. The S-curve distribution was originally defined to last exactly ten (10) years, and the contract would naturally stop minting once that period elapsed. After the fork to the new Block Reward contract, no new EWT will be minted to either block authors or the community fund. Instead of gradually phasing out over ten (10) years, the inflation will drop to zero (0) immediately at the chosen fork block and this action will effectively cap the circulating supply at a new lower value than the originally projected maximum.

2.2 In-Scope Components

The following change was in-scope for this assessment:

- *Disabling Rewards*

The relevant changes are located at the following URL:

- *"https://github.com/energywebfoundation/ewc-system-contracts/pull/92/files"*

Component	Commit Identifiers
"https://github.com/energywebfoundation/ewc-system-contracts/pull/92/files"	*67ac2f47b89c5fb3fa63dfdaa122708614276a3f* *ae5f32f2320c2a48aff38ede23606c9f96561943* *e7eb2311812ba67a9848996673d465072ca76501*

Other parts of the codebase that were not changed in this PR, were not included in the scope of this assessment.

A retesting phase was carried out on July 10th, 2025 targeting the merged master branch to validate the integrity and stability of the consolidated codebase:

Component	Commit Identifiers
"https://github.com/energywebfoundation/ewc-system-contracts"	*54eee51c0e2742bcb7b3f4c70600a2543677621b*

3 Methodology

3.1 Assessment Methodology

Chaintroopers' methodology attempts to bridge the penetration testing and source code reviewing approaches in order to maximize the effectiveness of a security assessment.

Traditional pentesting or source code review can be done individually and can yield great results, but their effectiveness cannot be compared when both techniques are used in conjunction.

In our approach, the application is stress tested in all viable scenarios though utilizing penetration testing techniques with the intention to uncover as many vulnerabilities as possible. This is further enhanced by reviewing the source code in parallel to optimize this process.

When feasible our testing methodology embraces the Test-Driven Development process where our team develops security tests for faster identification and reproducibility of security vulnerabilities. In addition, this allows for easier understanding and mitigation by development teams.

Chaintroopers' security assessments are aligned with OWASP TOP10 and NIST guidance.

This approach, by bridging penetration testing and code review while bringing the security assessment in a format closer to engineering teams has proven to be highly effective not only in the identification of security vulnerabilities but also in their mitigation and this is what makes Chaintroopers' methodology so unique.

The specific phase of the assessment was only limited to source code review.

3.2 Smart Contracts

The testing methodology used is based on the empirical study "Defining Smart Contract Defects on Ethereum" by J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo and T.

Chen, in IEEE Transactions on Software Engineering, and the security best practices as described in "Security Considerations" section of the solidity wiki.

The following is a non-exhaustive list of security vulnerabilities that are identified by our methodology during the examination of the in-scope contract:

- Unchecked External Calls
- Strict Balance Equality
- Transaction State Dependency
- Hard Code Address
- Nested Call
- Unspecified Compiler Version
- Unused Statement
- Missing Return Statement
- Missing Reminder
- High Gas Consumption Function Type
- DoS Under External Influence
- Unmatched Type Assignment
- Re-entrancy
- Block Info Dependency
- Deprecated APIs
- Misleading Data Location
- Unmatched ERC-20 standard
- Missing Interrupter
- Greedy Contract
- High Gas Consumption Data Type

4 Scoring System

4.1 CVSS

All issues identified as a result of Chaintroopers' security assessments are evaluated based on Common Vulnerability Scoring System version 3.1 (https://www.first.org/cvss/).

With the use of CVSS, taking into account a variety of factors a final score is produced ranging from 0 up to 10. The higher the number goes the more critical an issue is.

The following table helps provide a qualitative severity rating:

Rating	CVSS Score
None/Informational	0.0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

Issues reported in this document contain a CVSS Score section, this code is provided as an aid to help verify the logic of the team behind the evaluation of a said issue. A CVSS calculator can be found in the following URL:

https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

5 Identified Findings

5.1 Informational Findings

5.1.1 Lack of a dedicated upgrade mechanism

Description **INFO**

The Energy Web Chain's "*BlockReward*" implementation is designed as an immutable, non-upgradeable contract. Unlike proxy-based patterns (e.g., using an "*TransparentUpgradeableProxy" or "BeaconProxy"),* the "*BlockReward*" contract is referenced directly in the chain specification, and any change to the block-reward logic can only be implemented via a protocol-level (hard-fork) transition, and requires updating the spec to point to a new contract at a certain block.

As part of the security assessment, it was determined that the existing implementation does not employ any proxy or delegation mechanism (such as an *"EIP-1967*" implementation pointer, a fallback function performing "*delegateCall"*, or a proxy contract) and lacks any "*upgradeTo*" functionality. Furthermore, its storage layout is bound directly to this contract rather than being abstracted through a separate proxy contract. The planned update will introduce a new contract file ("*reward-0/BlockReward.sol*") containing the zero-reward logic, while preserving the original contract ("*reward/BlockReward.sol*") for archival reference.

In the absence of an on-chain upgrade mechanism, every validator and node operator must download and install a revised chain specification, and the network must mutually agree on a specific block height at which the transition will take effect.

Impact

Any miscommunication or delay in adopting the updated chain spec can result in some nodes continuing to call the old "*BlockReward*" contract while others have switched to the new one. This can cause inconsistent block-reward distributions, validator confusion, and, in extreme cases, a temporary chain split if blocks' validity rules start to diverge.

Finally, the directory name "*reward-0*" can be misleading, since "0" typically denotes an initial (or empty) version rather than a successor to the original.

Recommendation

A carefully coordinated hard-fork is essential when deploying and activating the new "*BlockReward*" contract via a chain-spec update. To ensure a smooth transition, it is imperative to:

- Select a block height sufficiently far in the future to allow all node operators adequate time to upgrade their software.
- Distribute the revised chain specification (or corresponding client release) and confirm that every validator has installed the update.
- Observe the designated fork block, verifying that no validator continues to execute the legacy reward logic.

Finally, it is recommended to prefer a more conventional naming approach based on a sequential or semantic-versioning scheme, such as:

- *"reward-1/BlockReward.sol"* (if upgrades start from 1)
- "*reward-v2/BlockReward.sol*" (Assuming that the new zero-reward logic is version 2)

CVSS Score

AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N/E:U/RL:X/RC:R/CR:X/IR:X/AR:X/MAV:X/MAC:X/MPR:X/MUI:X/MS:X/MC:X/MI:X/MA:X

5.1.2 Tests were not updated

Description **INFO**

The test suite for "*ewc-system-contracts*" was developed under the assumption of the original S-curve and constant-payout mechanisms; consequently, any test asserting a nonzero reward will no longer pass.

The issue exists at the files in the following location:

- *"https://github.com/energywebfoundation/ewc-system-contracts/tree/master/test/reward"*

```
File: https://github.com/energywebfoundation/ewc-system-
contracts/blob/master/test/reward/blockreward_function.js:132

calculateBlockReward(_blockNumber) {
        const bnBn = new this.web3.utils.BN(_blockNumber);
        if (bnBn.gte(this.maxBlockNumReward)) {
            return (new this.web3.utils.BN("0"));
        }
        return this.sCurve[bnBn.div(this.stepSize).toNumber(10)];
    }
```

Tests must be adjusted to expect zero rewards.

Impact

Any preexisting test cases will no longer succeed. Failing tests can yield an immediate interruption to CI/CD workflows:

- Automated pipelines relying on the current test suite will report failures, preventing merges until tests are updated.
- Any downstream tooling that interprets test outcomes will be disrupted.

Recommendation

It is advisable to update the test files as following:

- Any test that asserted that a block author's balance increases after a reward or that the community fund accrues tokens will need to be changed or removed.
- New tests should be added to verify that after the fork block, the reward() call returns the two addresses with 0 amounts, and that the internal counters don't change

For example, in file "*https://github.com/energywebfoundation/ewc-system-contracts/blob/master/test/reward/blockreward_function.js*" the function "*calculateBlockReward*" should be changed to:

```
calculateBlockReward(_blockNumber) {
    return new this.web3.utils.BN("0");
}
```

CVSS Score

AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N/E:U/RL:X/RC:R/CR:X/IR:X/AR:X/MAV:X/MAC:X/MPR:X/MUI:X/MS:X/MC:X/MI:X/MA:X

5.1.3 Dead code was not removed

Description **INFO**

It was identified that the implementation still inherits from the "*SCurveProvider*" library and retains state variables (e.g., "*communityFundAmount*") that are no longer utilized in any reward calculations. Similarly, the "*_logCommunityMinted*" and the "*_logMinted*" functions will be invoked with zero reward values. While this residual code does not pose a security risk, it may lead to confusion for future maintainers and slightly higher gas costs when fetching or deploying the code (e.g. *CREATE* on first deploy).

The issue exists at the following location:

```
File: https://github.com/energywebfoundation/ewc-system-
contracts/blob/e7eb2311812ba67a9848996673d465072ca76501/contracts/reward-
0/BlockReward.sol
import "../reward/SCurveProvider.sol";
import "../libs/SafeMath.sol";

/// @title Block reward contract
/// @notice Performs payouts at each new created block. Block authors
/// are rewarded according to an S-curve, while there is a constant
payout for
/// a community fund for a certain period of time.
/// @dev Contract is used by the Parity client and its address is
/// specified in the chainspec
contract BlockReward is SCurveProvider, IBlockReward {


    uint256 public communityFundAmount;
    /// Address of the community fund. Preferably a multisig wallet
    address public communityFund;
    /// Mapping of addresses and their payout addresses where rewards are
minted
    mapping(address => address) public payoutAddresses;

    constructor(address _communityFundAddress, uint256
_communityFundAmount)
        public
    {
        communityFund = _communityFundAddress;
        communityFundAmount = _communityFundAmount;
    }
```

It should be noted that the leftover code doesn't execute any harmful behavior so it's not introducing a vulnerability.

Impact

The leftover code may cause confusion for future maintainers and incur slightly higher gas costs during the first deployment (e.g., CREATE operations).

It should be noted that subsequent calls to the zero-reward functions will not incur gas for the S-curve logic, since those branches are never taken. Furthermore, this residual code does not introduce any security vulnerabilities.

Recommendation

It is advisable to:

- Remove or disable the *"SCurveProvider"* functions and the *"communityFundAmount"* variable
- Change the_*"checkRewardPeriodEnded"* function to always return true

CVSS Score

AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:N/E:U/RL:X/RC:R/CR:X/IR:X/AR:X/MAV:X/MAC:X/MPR:X/MUI:X/MS:X/MC:X/MI:X/MA:X

6 Retesting Results

6.1 Retest of Informational Findings

Two (2 out of 3) reported issues bearing no risk (INFORMATIONAL) were found to be sufficiently mitigated (*5.1.1, 5.1.2)*. One (1 out of 3) INFORMATIONAL issue was also marked as risk-accepted (see section *5.1.3*).

6.1.1 Lack of a dedicated upgrade mechanism

A comprehensive protocol-level node upgrade guide has been released to streamline the coordination of node transitions during the planned hard fork, and to avoid any inconsistent block-reward distributions or validator confusion. The guide can be found in the following URL:

https://github.com/energywebfoundation/ewf-zurich-upgrade

Stakeholders are strongly encouraged to follow the guide meticulously to ensure a smooth and deterministic migration process.

Furthermore, a naming approach based on a semantic-versioning scheme was introduced to replace the original misleading directory name:

```
Commit: 42c9625e0ef83c3c3610867b21c2625a8178b3eb
contracts/reward-0/BlockReward.sol renamed to contracts/reward-
v2/BlockReward.sol
```

As a result, the issue was marked as CLOSED.

6.1.2 Tests were not updated

The team confirmed that no additional forks are planned and that no continuous integration/continuous deployment (CI/CD) pipeline relies on these tests. Consequently, the issue was marked as CLOSED.

6.1.3 Dead code was not removed

This decision was made deliberately, reflecting the team's intent to minimize modifications to the original contract and preserve its existing structure and logic.

As a result, the issue was marked as risk-accepted.

References & Applicable Documents

Ref.	Title	Version
N/A	*N/A*	*N/A*

Document History

Revision	Description	Changes Made By	Date
0.5	*Initial Draft*	*Chaintroopers*	*June 5th, 2025*
1.0	*First Version*	*Chaintroopers*	*June 6th, 2025*
1.1	*Added Retest Results*	*Chaintroopers*	*July 10th, 2025*