WEICHAIN Security Assessment Certificate

26th of October 2023 — WeiChain Verified

Epoch Island

This smart contract audit was prepared by WeiChain.

Post-Audit Conclusion

The **Epoch Island** team remediated all exhibits outlined in the report. The codebase adheres to the best practices and standards of smart contract development. During the audit process, our team thoroughly reviewed the smart contract code, conducted various tests, and analyzed the contract's functionality and security aspects. We initially identified several areas that required attention, including potential vulnerabilities and code inefficiencies. However, we are pleased to note that all these issues have been successfully resolved and acknowledged.

The improvements made to the smart contract code are commendable. The codebase demonstrates a robust structure, with clear and concise logic that is understandable. The contract's functionality has been thoroughly tested, ensuring that it performs as intended and meets the specified requirements. Furthermore, the contract implements essential security measures, mitigating potential risks and vulnerabilities.

The codebase can be considered of a good quality. It exhibits a high level of clarity, with well-commented sections and consistent naming conventions, making it easy to maintain and understand. The use of standardized libraries and frameworks has enhanced code readability and reduced the likelihood of introducing bugs or vulnerabilities.

Overall, we are confident in stating that the smart contract of **Epoch Island**, has successfully passed the post-audit evaluation. The codebase is of good quality, following good practices, and all identified issues have been resolved. We commend your commitment to ensuring a secure and well-developed smart contract.

Executive Summary

Type	Smart Contracts
Auditors	Krasimir Raykov
Timeline	2023-10-10 through 2023-10-26
EVM	Shanghai
Languages	Solidity
Methods	Architecture Review, Computer-Aided Verification, Manual Review
Specifications	README.md

Total issues	8
Critical	0
Medium	0
Low	6
Informational	2

Critical	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for clients and users.
Medium	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.

Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

The scope of the audit is restricted to the set of files outlined in the Audit Breakdown section. While reviewing the given files we identified **0 issue of critical severity**, **0 issues of medium severity**, **5 issue of low severity and 2 issues of informational severity.**
In addition, we made several suggestions with regards to code documentation, adherence to best practices, and adherence to the specification. We recommend resolving the issues and improving code documentation before shipping to production.

ID	Description	Severity	Status
WCH – 1	Compiler version not fixed	Low	Resolved
WCH – 2	Overpowered role	Low	Acknowledged
WCH – 3	Wrong usage of OpenZeppelin's ERC20 contract	Low	Acknowledged
WCH – 4	Phishing attack opportunity	Informational	Acknowledged
WCH – 5	Circulating Supply Impact	Informational	Acknowledged
WCH – 6	Denial of service	Low	Resolved
WCH – 7	Precision lost	Low	Acknowledged
WCH – 8	Reentrancy events	Low	Resolved

Code Coverage and inline documentation

The contracts that are in-scope are well covered with tests. We strongly advise the implementation of comprehensive unit tests, with the goal of achieving a minimum code coverage of 90%

Static Analysis

The execution of our static analysis toolkit identified **214 potential issues** within the codebase of which **202 were ruled out to be false positives** or negligible findings.

The remaining **issues** were validated and grouped and formalized into the **1 exhibits** - WCH – 7

Audit Breakdown

WeiChain's objective was to evaluate the following files for security-related issues, code quality, and adherence to specification and best practices:

* EpochToken.sol
* EpochAirdrop.sol
* Vepoch.sol
* EpochUpsidePoolV1.sol

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The WeiChain auditing process follows a routine series of steps:

1. Code review that includes the following
    1.1. Review of the specifications, sources, and instructions provided to WeiChain to make sure we understand the size, scope, and functionality of the smart contract.
    1.2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    1.3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to WeiChain describe.
2. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
3. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Findings

WCH-1 Compiler version not fixed

Severity: **Low**

Status: Resolved

File(s) affected: All

Description: Solidity source files indicate the versions of the compiler they can be compiled with. It is recommended to specify the exact compiler version (pragma solidity x.y.z;), as future compiler versions may handle certain language constructions in a way the developer did not foresee.

Recommendation: Use a specific compiler version. Example:

```
pragma solidity 0.8.21;
```

WCH-2 Overpowered role

Severity: **Low**

Status: Acknowledged

File(s) affected: EpochToken.sol

Description: There are functions callable only from one address. Therefore, the system depends heavily on this address. In this case, there are scenarios that may lead to undesirable consequences for investors.

WCH-3 Wrong usage of OpenZeppelin's ERC20 contract

Severity: Low

Status: Acknowledged

File(s) affected: Vepoch.sol

Description: transfer and transferFrom requires authorization. To achieve that one must override the virtual method **_beforeTokenTransfer** instead of overriding the transfer and transferFrom methods. The reason for that is this could lead to undesired changes to the transfer functionality.

Recommendation: See the following example: https://docs.openzeppelin.com/contracts/4.x/extending-contracts#using-hooks

WCH-4 Phishing attack opportunity

Severity: **Informational**

Status: Acknowledged

File(s) affected: Vepoch.sol

Description: Without whitelisting, there is the potential for someone to create a deceptive token and employ it as an "upsideToken" while expecting a legitimate "downsideToken" in return. When I refer to a "malicious" token, I mean a situation where, for instance, the transfer function fails to genuinely transfer funds or only does so selectively in specific cases.

Recommendation: There are a couple of remedies for this:

1. The initial solution involves implementing token whitelisting on-chain or off-chain (frontend).

2. The second approach is to perform balance checks on the contract after each transfer, similar to what UniswapV2 does.

Although the probability of such a situation occurring is relatively low, it could serve as an opportunity for a potential phishing attack.

WCH-5 Circulating Supply Impact

Severity: Informational

Status: Acknowledged

File(s) affected: EpochToken.sol

Description: Circulating Supply refers to the number of coins or tokens of a specific cryptocurrency that are publicly available to buy or sell. If you can trade them, they are considered circulating. The more coins are added to circulation, the more the value decreases. Conversely, the more coins are burned or removed from circulation, the more the value increases. In the EpochToken, the total supply **could be** increased by 10% each week by the token owner which in that case is a multisig (safe). Starting at 100M total supply it could take 70 days to double the supply. After 1 year the total supply **could reach** ~3,2B tokens.

WCH-6 Denial of service

Severity: **Low**

Status: Resolved

File(s) affected: EpochUpsidePoolV1.sol

Description: The start date should not be after than the end date and that's currently possible. This falls under the DoS (denial of service) attacks, because if the start date is greater than the end date, then the take function will always revert because of this method:

```
uint256 percentageFee = ((d.endDate - block.timestamp) * (10**18)) / (d.endDate - d.startDate);
```

Recommendation: Check that end date is greater than the start date in the `supply` method.

WCH-7 Precision lost

Severity: **Low**

Status: Acknowledged

File(s) affected: EpochUpsidePoolV1.sol

Description: Solidity's integer division truncates. Thus, performing division before multiplication can lead to precision loss.

Recommendation: Consider ordering multiplication before division.

WCH-8 Reentrancy events

Severity: **Low**

Status: Resolved

File(s) affected: Vepoch.sol

Description: Reentrancy that allow manipulation of the order of events. This may cause issues for offchain components that rely on the values of events.

Recommendation: Apply the check-effects-interactions pattern.

Disclaimer

WeiChain audit is not a security warranty, investment advice, or an endorsement of EpochIsland or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of WeiChain from legal and financial liability.

WeiChain Ltd.