Blaize.Security

WEPC
SMART CONTRACT AUDIT

July 18th, 2024 / V. 1.0

Table of Contents

Executive Summary

During the audit, we examined the security of smart contracts for the **WEPC** protocol. Our task was to find and describe any security issues in the smart contracts of the platform. This report presents the findings of the security audit of the **WEPC** smart contracts conducted between **July 15th, 2024 and July 17th, 2024.**

The Blaize.Security team audited the WEPC protocol, an ERC20-compatible token enhanced with the burning and minting interfaces. The Protocol Overview section displays more information about the contract logic.

The security team verified the correctness of the ERC20 interface implementation, checked the integrity of the token logic, and the access control system. Checks included the standard checklist and manual review by the auditors. The team noted 2 areas which produce increased risks for the system - both are connected with the minting and burning processes. All issues are described in the Complete Analysis section. The WEPC team verified that the implemented logic corresponds to the desired business logic.

Summary notes:

1. The security team verifies that the WEPC token implements the ERC20 interface correctly and according to the standard.
2. The contract uses the OpenZeppeling standard contracts of the v5.0.2 release. However, the team should note that although the contract implements the standard library, some of the inherited libraries and interfaces are not used, and the contract size has just increased. This is not a security issue, as it relies on the OpenZeppelin implementation, which can be suboptimal but secure.
3. The token implements the public burn() / burnFrom() interface via the OZ ERC20Burnable interface. While this is standard practice, the Customer should be aware of the potential risks associated with it, listed in the Info-1 issue.

4. The token implements unrestricted mint functionality. While it creates risks of its own (listed in the Info-2 issue), the correct access control and sanitizing measures usually minimize such risks. However, based on the current state of the contract's settings, the Admin and Minter roles are assigned to the same address, thus creating a single point of failure risk—having a negative impact on the security. In this case, a single compromised key will compromise both of the authorized roles.
5. Token implements the ERC-2612 Permit interface. While it is a regular practice, it requires additional monitoring of its usage and the cautious behavior of users. Permit functionality creates several potentially dangerous situations for users:
   - the signed permit cannot be withdrawn; thus it exists until it is either used or re-written by the alternative signed permit
   – users should validate the deadline for the signature, as in the case of the unvalidated considerable number, it will exist until executed (as it cannot be withdrawn)
   - several simultaneous signatures may exist for the same nonce, creating a racing condition
   Thus, users and the Customer should be aware of these conditions and inform the community accordingly.
6. No initial Hardhat/Foundry project was provided, so the security team cannot evaluate the deployment and testing approaches and their potential impact on further development.

Therefore, the security team verified the ERC20 implementation and evaluated the project as **Secure**. Thus, the WEPC project successfully passed the audit, though auditors noted the list of potential risks acknowledged by the WEPC team.

Auditing strategy and Techniques applied/Procedure

Blaize.Security auditors start the audit by developing an auditing strategy - an individual plan where the team plans methods, techniques, approaches for the audited components. That includes a list of activities:

MANUAL AUDIT STAGE

- Manual line-by-line code by at least 2 security auditors with crosschecks and validation from the security lead;
- Protocol decomposition and components analysis with building an interaction scheme, depicting internal flows between the components and sequence diagrams;
- Business logic inspection for potential loopholes, deadlocks, backdoors;
- Math operations and calculations analysis, formula modeling;
- Access control review, roles structure, analysis of user and admin capabilities and behavior;
- Review of dependencies, 3rd parties, and integrations;
- Review with automated tools and static analysis;
- Vulnerabilities analysis against several checklists, including internal Blaize.Security checklist;
- Storage usage review;
- Gas (or tx weight or cross-contract calls or another analog) optimization;
- Code quality, documentation, and consistency review.

and a wide spectrum of other vulnerable areas.

FOR ADVANCED COMPONENTS:

- Cryptographical elements and keys storage/usage audit (if applicable);
- Review against OWASP recommendations (if applicable);
- Blockchain interacting components and transactions flow (if applicable);
- Review against CCSSA (C4) checklist and recommendations (if applicable);

TESTING STAGE:

- Development of edge cases based on manual stage results for false positives validation;
- Integration tests for checking connections with 3rd parties;
- Manual exploratory tests over the locally deployed protocol;
- Checking the existing set of tests and performing additional unit testing;
- Fuzzy and mutation tests (by request or necessity);
- End-to-end testing of complex systems;

In case of any issues found during audit activities, the team provides detailed recommendations for all findings.

POST-AUDIT STEPS RECOMMENDED

To ensure the security of the contract, the **Blaize.Security** team suggests that the **WEPC** team follow post-audit steps:

1. Request audits of other protocol components (dApp, backend, wallet, blockchain, etc) from Blaize Security
2. Request consulting and deployment overwatch services provided by Blaize Security
3. Launch active protection over the deployed contracts to have a system of early detection and alerts for malicious activity. We recommend the AI-powered threat prevention platform **VigiLens**, by the **CyVers** team.
4. Launch a **bug bounty program** to encourage further active analysis of the smart contracts.
5. Request post-deployment assessment service provided by Blaize Security to ensure the correctness of the configuration, settings, cross-connections of deployed entities and live functioning

Audit Rating

Score: 9.7/10

	RATING
Security	9.7
Logic optimization	9.9
Code quality	10
Testing suite	0
Documentation	N/A

Security: General mark for the security of the protocol.
The main mark for the audit qualification.

Logic optimization: Evaluation of how optimal the implementation is, including presence of extra/unused code, uncovered/extra cases, gas (or its analog) optimization, memory management optimization, etc
Code quality: Evaluation of best practices followed, code readability, structure and convenience of further development
Testing suite: Availability of the native tests suite, level of logic coverage, checks of critical areas being covered.
Documentation: Availability and quality of the documentation, coverage of core functionality and user flows: whitepaper, gitbook, readme, specs, natspec, comments in the code and other possible forms of documentation.

SECURITY RATING CALCULATION

Approximate weight of unresolved issues.
Critical: -3 points
High: -2 points
Medium: -0.5 points
Low: -0.1 points
Informational: -0.1 point (in general, depends on the context)

Note: additional concerns, violated checklist items (including standard vulnerabilities), and verified backdoors may influence the final mark and weight of certain issues.

Starting with a perfect score of 10:

Critical issues: 0 issue (0 resolved): 0 points deducted

High issues: 0 issues (0 resolved): 0 points deducted

Medium issues: 0 issue (0 resolved): 0 points deducted

Low issues: 0 issues (0 resoled): 0 points deducted

Informational issues: 2 issues (2 verified): -0.1 points deducted based on the concern of the unrestricted minting, which fails the Access Control check

Other:
-0.1 points deducted for the failed check against the single point of failure risk

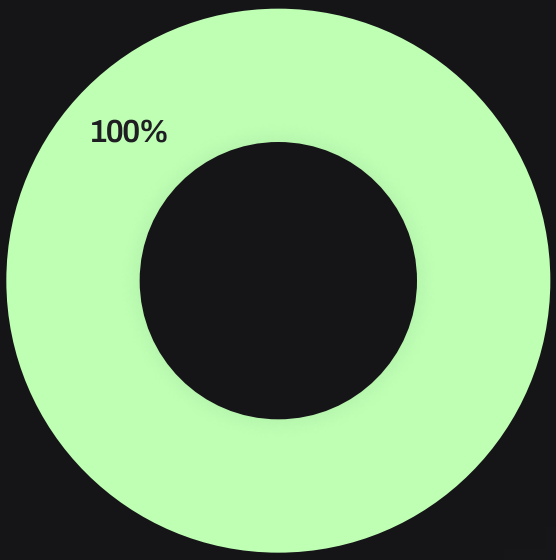
-0.1 points deducted for the absence of the project thus failing the check of the tests suite, development approach and organization.

Security rating = 10 -0.1 - 0.2= 9.7

Technical Summary

THE GRAPH OF VULNERABILITIES DISTRIBUTION:

- Critical
- High
- Medium
- Low
- Info



The table below shows the number of the detected issues and their severity. A total of 19 problems were found. 18 issues were fixed or verified by the Customer's team.

	FOUND	FIXED/VERIFIED
Critical	0	0
High	0	0
Medium	0	0
Low	0	0
Info	2	2

SEVERITY DEFINITION

CRITICAL

The system contains several issues ranked as very seriousand dangerous for users and the secure work of the system. Requires immediate fixes and a further check.

HIGH

The system contains a couple of serious issues, which lead to unreliable work of the system and migh cause a huge data or financial leak. Requires immediate fixes and a further check.

MEDIUM

The system contains issues that may lead to medium financial loss or users' private information leak. Requires immediate fixes and a further check.

LOW

The system contains several risks ranked as relatively small with the low impact on the users' information and financial security. Requires fixes.

INFO

The issue has no impact on the contract's ability to operate, yet is relevant for best practices. Or this status can be assigned to the issues related to the suspicious activity or substandard business logic decisions which cannot be classified without the comments from the team (and can be re-classified on the later audit stages).

Issues reviewed by the team can get the next statuses:
Resolved: issue is resolved by an appropriate patch or changes in the business logic
Verified: the team provided sufficient evidences that the issue describes desired behavior
Unresolved: neither path nor comments provided by the team, or they are not sufficient to resolve the issue
Acknowledged: the team accepts the misbehavior and connected risks

Audit Scope

Language/Technology: **Solidity**

Blockchain: **Ethereum**

The scope of the project includes:

- WorldEarnPlayCommunity.sol

The source code of the smart contract was delivered in a form of the deployed contract:

- 0x94b2d91aa89ba30f2e080205c14e83fcdee310dd

Protocol overview

DESCRIPTION

WorldEarnPlayCommunity is a token implementing the ERC20 interface with several extensions:

1. The token implements the OZ Burnabble interface
2. The token implements the ERC-2612 Permit interface (OZ implementation)
3. The token implements the standard OZ AccessControl roles model to define the Minter role

Token has:

- 18 decimals
- WEPC ticker
- "World Earn - Play Community" name
- 0 initial supply
- Minter-role responsible for the further token minting with no supply cap

ROLES AND RESPONSIBILITIES

1. Admin:

- Defined as DEFAULT_ADMIN_ROLE (standard OZ AccessControl role)
- Can set and renounce other roles in the contract
- Is set during the contract construction to the address defined in the parameter

2. Minter

- Defined as MINTER_ROLE (via the OZ AccessControl approach)
- can mint any amount of tokens to any user at any time with no restrictions.
- Is set during the contract construction to the address defined in the parameter (separate from the ADMIN)

CONFIGURATION AND SETTINGS

1. Admin
Currently set to: 0xaf085641b044566Fc2367A49A6BFa73179759DFe

2. Minter
Currently set to: 0xaf085641b044566Fc2367A49A6BFa73179759DFe

LIST OF VALUABLE ASSETS

1. WEPC token itself

- the Minter role controls token minting
- anyone can burn the token from their addresses
- user can burn tokens from other users in case of the appropriate allowance

No tokens are supposed to be on the contract's balance. Therefore, the contract does not support rescue functionality, so any tokens sent directly to the contract will be locked.

DEPLOYMENT

The contract is already deployed on the Ethereum mainnet:
https://etherscan.io/token/0x94b2d91aa89ba30f2e080205c14e83fcdee310dd#code

No original Hardhat/Foundry project was provided; thus, the initial deployment scripts were not reviewed.

Complete Analysis

STANDARD CHECKLIST / VULNERABLE AREAS

Storage structure and data modification flow	Pass
Access control structure, roles existing in the system	Pass
Public interface and restrictions based on the roles system	Failed
General Denial Of Service (DOS)	N/A
Entropy Illusion (Lack of Randomness)	N/A
Order-dependency and time-dependency of operations	Pass
Accuracy loss, incorrect math/formulas other violated operations with numbers	Pass
Validation of function parameters, inputs validation	Pass
Asset management, funds flow and assets conversions	N/A
Signatures replay and multisig schemes security	N/A
Asset Security (backdoors connected to underlying assets)	Pass
Incorrect minting, initial supply or other conditions for assets issuance	Pass
Global settings mis-using, incorrect default values	Pass
Violated communication between components/modules, broken co-dependencies	N/A
3rd party dependencies, used libraries and packages structure	N/A
Single point of failure	Failed
Centralization risk	Pass
General code structure checks and correspondence to best practices	Pass
Language-specific checks	Pass

DISCOVERED ISSUES

INFO-1	Verified

NO RESTRICTIONS ON THE TOKEN BURNING

The contract inherits the default ERC20Buranble interface, which contains public burn() and burnFrom() functions
While it is a regular functionality that allows users to burn tokens from their accounts, it still possesses several risks to the tokens economy:

- intentional decreasing of the supply by malicious actors
- secondary threats from the protocols holding the token - in case of their exploit
- secondary threats from burning of tokens via “dangling approves.”

The issue is marked as Info—while such functionality creates certain risks, it is a regular token extension. However, the security team requires additional verification from the Customer about the awareness of existing risks connected to this business logic decision.

RECOMMENDATION:

Verify the protocol's unrestricted burn() functionality is desired and potential risks are acknowledged.

POST-AUDIT

The Customer verified the burn logic as desired and acknowledged the potential risks.

INFO-2 Verified

UNRESTRICTED TOKEN MINTING

The token implements the mint() interface controlled by the Minter role. However, there are no restrictions on the minting process:

- Minter can mint any amount of tokens, at any moment of time, for any user
- no restrictions on the total supply of the token; thus it can grow to any value

While the flexible supply and controlled minting are regular models, the absence of control measures creates several risks:

- centralization risk, as the Minter has unrestricted access to the supply
- secondary risks for the token economy in case of uncontrolled emission
- human factor risk - tokens can be minted to unvalidated addresses.

The issue is marked as Info—while such functionality creates certain risks, it is a regular token extension. However, the security team requires additional verification from the Customer about the awareness of existing risks.

RECOMMENDATION:

1) Verify that the protocol's unrestricted minting functionality is desired and potential risks are acknowledged - risks of minting to wrong addresses, risks of minting incorrect amounts, risk of increasing the inflation speed, risk of causing price volatility after certain amounts minting, etc.
2) Provide the sanitizing measures for the minting process: validation of Minter's assigned, validation of minted token receivers, validation of the minting schedule, control over the emission rate, and supply growth speed.
3) The better way is to encode restrictions into the smart contract: limits on one-time minting, supply cap or restrictions on the emission rate, and restrictions on the receiving addresses. Though the auditors recognize that it will require token re-deployment, so recommendations aim at the awareness of potential risks.

POST-AUDIT

The Customer verified the burn logic as desired and acknowledged the potential risks. However, the security team leaves the concern regarding the risks caused by the absence of minting restrictions.

Disclaimer

The information presented in this report is an intellectual property of the customer, including all the presented documentation, code databases, labels, titles, ways of usage, as well as the information about potential vulnerabilities and methods of their exploitation. This audit report does not give any warranties on the absolute security of the code. Blaize.Security is not responsible for how you use this product and does not constitute any investment advice.

Blaize.Security does not provide any warranty that the working product will be compatible with any software, system, protocol or service and operate without interruption. We do not claim the investigated product is able to meet your or anyone else's requirements and be fully secure, complete, accurate, and free of any errors and code inconsistency.

We are not responsible for all subsequent changes, deletions, and relocations of the code within the contracts that are the subjects of this report.

You should perceive Blaize.Security as a tool, which helps to investigate and detect the weaknesses and vulnerable parts that may accelerate the technology improvements and faster error elimination.