#HashEx
BLOCKCHAIN SECURITY

ENO

smart contracts
final audit report

July 2024

hashex.org

contact@hashex.org

Contents

1. Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and HashEx and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (HashEx) owe no duty of care towards you or any other person, nor does HashEx make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and HashEx hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, HashEx hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against HashEx, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed. HashEx owns all copyright rights to the text, images, photographs, and other content provided in the following document. When using or sharing partly or in full, third parties must provide a direct link to the original document mentioning the author (hashex.org).

2. Overview

HashEx was commissioned by the ENO team to perform an audit of their smart contract. The audit was conducted between 23/07/2024 and 25/07/2024.

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

The code is available in the @ENOToken/Badge-ENO-Contracts GitHub repository after the commit 846f24f.

Update. A recheck was done after the commit b850e1e.

Important: Verification of the deployed smart contracts was not performed. Users who interact with the contracts must verify themselves that the deployed code is identical to the audited code.

2.1 Summary

Project name	ENO
URL	https://enotoken.io/
Platform	Arbitrum Network
Language	Solidity
Centralization level	● Low
Centralization risk	● Low

2.2 Contracts

Name	Address
NFTENO	

3. Project centralization risks

C20CR2b Owner privileges

The contract owner can update token metadata only before the sale starts.

4. Found issues

4
Total issues

●	Low	1 (25%)
●	Info	3 (75%)

C20. NFTENO

ID	Severity	Title	Status
C20I05	● Low	Gas optimizations	✓ Resolved
C20I06	● Info	Typo in variable name	✓ Resolved
C20I07	● Info	Inconsistent usage of SafeERC library to transfer tokens	✓ Resolved
C20I08	● Info	No cap for NFT price	✓ Resolved

5. Contracts

C20. NFTENO

Overview

NFTENO is a non-fungible token of ERC721 standard made with ERC721Enumerable implementation from OpenZeppelin contract library. Minting is a public function, requiring user to complete payment in a single form of ERC20 ENO tokens.

Issues

C20I05 Gas optimizations ● Low ✓ Resolved

1. The `commissionWallet`, `ownerWallet`, `enoToken`, `saleStartTime`, `maxMintsPerWallet`, `sameMetadataForAll`, `comision` variables can be declared as immutable.

2. Multiple reads from storage in the `buyNFTWithENO()` function: `NFTPriceInENO`, `enoToken`, `_mintedCount[msg.sender]` variables.

3. 3 transfers of ERC20 tokens are made instead of 2 are made in the `buyNFTWithENO()` function.

4. Multiple reads from storage in the `mint()` function: `_tokenId` variable.

Update

The issue was fixed in commits 29f08c6, b6c54ba, 96e5372.

C20I06 Typo in variable name ● Info ✓ Resolved

The contract state variable `comision` should be renamed to `commission`.

Update

The issue was fixed in commit 4ee0547.

C20I07 Inconsistent usage of SafeERC library to transfer tokens ● Info ✓ Resolved

The `buyNFTWithENO` function in the `NFTENO` contract uses the `transferFrom` method from ENO's token contract to transfer tokens to the `NFTENO` contract. Later in the same function, tokens are transferred using the `safeTransferFrom`method. It is recommended to use the `safeTransferFrom` function consistently throughout the contract for token transfers to ensure compatibility and security. The `safeTransferFrom` method from OpenZeppelin's `SafeERC20` library correctly handles various implementations of ERC20 tokens, including those that do not return a boolean value.

```
function buyNFTWithENO() public nonReentrant {
        ...
        require(enoToken.transferFrom(msg.sender, address(this), NFTPriceInENO), "Failed
to transfer ENO");

        ...
        enoToken.safeTransfer(commissionWallet, commissionAmount);
        enoToken.safeTransfer(ownerWallet, ownerAmount);
        ...
}
```

Update

The issue was fixed in commit b850e1e.

C20I08 No cap for NFT price ● Info ✓ Resolved

The function `setNFTPriceInENO()` function allows the owner to set any price for the NFTs, which could potentially be exploited to set an extremely high price.

We recommend to set a reasonable cap for the maximum NFT price.

Update

The issue was fixed in commit f60907b. The function `setNFTPriceInENO()` was removed.

6. Conclusion

1 low severity issue was found during the audit. 1 low issue was resolved in the update. The reviewed contracts are highly dependent on the owner's account. See the centralization risks chapter.

Important: Verification of the deployed smart contracts was not performed. Users who interact with the contracts must verify themselves that the deployed code is identical to the audited code.

Appendix A. Issues' severity classification

- **Critical.** Issues that may cause an unlimited loss of funds or entirely break the contract workflow. Malicious code (including malicious modification of libraries) is also treated as a critical severity issue. These issues must be fixed before deployments or fixed in already running projects as soon as possible.
- **High.** Issues that may lead to a limited loss of funds, break interaction with users, or other contracts under specific conditions. Also, issues in a smart contract, that allow a privileged account the ability to steal or block other users' funds.
- **Medium.** Issues that do not lead to a loss of funds directly, but break the contract logic. May lead to failures in contracts operation.
- **Low.** Issues that are of a non-optimal code character, for instance, gas optimization tips, unused variables, errors in messages.
- **Informational.** Issues that do not impact the contract operation. Usually, informational severity issues are related to code best practices, e.g. style guide.

Appendix B. Issue status description

Resolved. The issue has been completely fixed.

Partially fixed. Parts of the issue have been fixed but the issue is not completely resolved.

Acknowledged. The team has been notified of the issue, no action has been taken.

Open. The issue remains unresolved.

Appendix C. List of examined issue types

- Business logic overview
- Functionality checks
- Following best practices
- Access control and authorization
- Reentrancy attacks
- Front-run attacks
- DoS with (unexpected) revert
- DoS with block gas limit
- Transaction-ordering dependence
- ERC/BEP and other standards violation
- Unchecked math
- Implicit visibility levels
- Excessive gas usage
- Timestamp dependence
- Forcibly sending ether to a contract
- Weak sources of randomness
- Shadowing state variables
- Usage of deprecated code

Appendix D. Centralization risks classification

Centralization level

- 🔴 **High.** The project owners can manipulate user's funds, lock user's funds on their will (reversible or irreversible), or maliciously update contracts parameters or bytecode.
- 🟠 **Medium.** The project owners can modify contract's parameters to break some functions of the project contract or contracts, but user's funds remain withdrawable.
- 🟢 **Low.** The contract is trustless or its governance functions are safe against a malicious owner.

Centralization risk

- 🔴 **High.** Lost ownership over the project contract or contracts may result in user's losses. Contract's ownership belongs to EOA or EOAs, and their security model is unknown or out of scope.
- 🟠 **Medium.** Contract's ownership is transferred to a contract with not industry-accepted parameters, or to a contract without an audit. Also includes EOA with a documented security model, which is out of scope.
- 🟢 **Low.** Contract's ownership is transferred to a well-known or audited contract with industry-accepted parameters.

contact@hashex.org

@hashex_manager

blog.hashex.org

linkedin

github

twitter

HashEx

BLOCKCHAIN SECURITY