VoidFarm

smart contract audit report

Prepared for:

VoidFarm

Authors: HashEx audit team

April 2021

Contents

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as 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.

DISCLAIMER: 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.

Introduction

HashEx was commissioned by the VoidFarm team to perform an audit of VoidFarm smart contracts. The audit was conducted between April 12 and April 16, 2021.

The audited code is located in VoidFarm's github repository [1]. The audit was performed after the commit `508c815`. A recheck was done after commit `1cc0311` [5]. There was limited documentation available at voidfarm.gitbook.io.
Update: VoidToken deployed to BSC at `0x3C44eAf8b4eAEF6e48Bfc18Ee92412BE0b395746` and MasterChef at `0xD72fF7178fb11141492Da457A1B3c4D5143b696c` are identical to reviewed [5] versions.

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 to understand 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.

Contracts overview

`Address.sol`

Similar to OpenZeppelin version of release v3.3 with pragma fixed to 0.6.12.

`Context.sol`

Similar to OpenZeppelin version of release v3.0 with pragma changed to 0.6.12.

`Ownable.sol`

Similar to a mix of OpenZeppelin's v2.5 and v3.0 with pragma changed to 0.6.12.

`ReentrancyGuard.sol`

Similar to OpenZeppelin version of release v3.1 with pragma fixed to 0.6.12.

`SafeMath.sol`

Similar to OpenZeppelin version of release v3.1 with pragma changed to 0.6.12.

`Timelock.sol`

Similar to Compound's version with minor changes. Audited [2] by OpenZeppelin in 2019.

`IBEP20.sol`

Similar to Binance's version with pragma changed to 0.6.12.

`SafeBEP20.sol`

Similar to OpenZeppelin version of release v3.3 with pragma changed to 0.6.12.

`Migrations.sol`

Migrations contract from Truffle project, unused.

`Multicall.sol`

Helper contract with functions for frontend.

`VoidToken.sol`

Implementation of BEP20 token with custom functionality.

`MasterChef.sol`

Similar to SushiSwap's chef contract with modifications.

Found issues

ID	Title	Severity	Response
01	BEP20 standard violation	High	Fixed
02	Lack of safeguards	Medium	Fixed
03	Unlimited mint by owner	Medium	Informed
04	Inconsistent mine rate	Medium	Fixed
05	Maximum supply can be exceeded	Medium	Fixed
06	Low severity issues & recommendations	Low	Fixed/Informed

#01 BEP20 standard violation (restriction of zero amount transfer) High

Implementation of `transfer()` function in VoidToken.sol does not allow to input zero transfer amount as it's demanded in ERC-20 [3] and BEP-20 [4] standards. This issue may break the interaction with smart contracts that rely on full ERC20 support.

Update: VoidFarm team has removed require statement for transfer amount > 0 in commit `1cc0311` [5].

#02 Lack of safeguards Medium

`dev()` and `setFeeAddress()` functions in MasterChef.sol should require non-zero input addresses.

Update: checks on non-zero addresses were added in commit `1cc0311` [5].

#03 Unlimited mint by owner Medium

`mint()` function in VoidToken.sol could be used by the owner of the contract to unlimitedly mint new tokens. However, the logic of the MasterChef.sol demands ownership of the VOID token. We recommend transferring ownership to MAsterChef contract as soon as possible after contract deployment.
Update: ownership was transferred to the MasterChef contract in
`0xfcf5e929e8d6b9bbd6457cc84fb39da1daeaa539e9618e7541c5464c5a37764d`.

#04 Inconsistent mine rate — Medium

Documentation on VoidFarm website claims that 0.01 Void mined per block. Around 288 Void tokens per day, a very low emission. Actually, in MasterChef.sol a bigger amount is mined per block as 0.01 Void is minted on the contract's balance and then 0.0002 Void are minted to dev's address. Moreover, the minted amount per block can be adjusted by the owner at any time (capped by 10 Void per block).

Update: issue was fixed in commit `1cc0311` [5].

#05 Maximum supply can be exceeded — Medium

Documentation on VoidFarm website claims that the max supply of Void is restricted by 30k. However, it could surpass restriction as the reward update could take any amount unless the current supply is less than 30k.

Update: issue was fixed in commit `1cc0311` [5]. It must be noted that with these changes if the token supply is bigger than `maxSupply` parameter (for example, more tokens were preminted) the MasterChef contract won't work because the function `updatePool` will always fail.

#06 Low severity and general recommendations — Low

1. VoidToken.sol L28-29 contains variables `_taxFee` and `_burnFee` that should be declared constans. **Update:** issue was fixed in [5].
2. MasterChef.sol L55 contain maxSupply variable should be declared constant. **Update:** issue was fixed in [5].
3. Documentation on VoidFarm website states a dev fee of 2%. Actually, it's slightly less (2 of 102). **Update:** issue was fixed in [5].
4. Any accidental direct, not via `deposit()`, Void token deposits to MasterChef.sol will be burned with the next `withdraw()` event.
5. Typo in a comment in MasterChef.sol L225. **Update:** issue was fixed in [5].
6. We recommend adding documentation to the smart contracts.
7. We recommend using tests before deployment. At least VoidToken and MasterChef should be covered.

Conclusion

It is crucially important for users before using the token to check that ownership of VoidToken is transferred to MasterChef contract as farming via `mint()` won't work and before this transfer owner of the token can mint an unlimited number of tokens.

One high severity issue was found regarding BEP20 token standard violation.

Audit includes recommendations on the code improving and preventing potential attacks.

Update: high severity issue was fixed before deployment amongst various fixes of the issues from the initial report.

Update: contracts deployed to BSC at `0x3C44eAf8b4eAEF6e48Bfc18Ee92412BE0b395746` and `0xD72fF7178fb11141492Da457A1B3c4D5143b696c` are identical to the reviewed ones. VoidToken's ownership was transferred to the MasterChef contract in `0xfcf5e929e8d6b9bbd6457cc84fb39da1daeaa539e9618e7541c5464c5a37764d`.

References

1. VoidFarm github repository
2. Timelock audit
3. ERC-20 standard
4. BEP-20 standard
5. VoidFarm fixes commit

Appendix. Issues' severity classification

We consider an issue critical if it may cause the unlimited losses or breaks the workflow of the contract and could be easily triggered.

High severity issues may lead to the limited losses or break interaction with users or other contracts under very specific conditions.

Medium severity issues do not cause the full loss of functionality, but break the contract logic.

Low severity issues are typically nonoptimal code, unused variables, errors in messages. Usually these issues do not need immediate reactions.