#HashEx
BLOCKCHAIN SECURITY

Onchain

smart contracts final audit report

May 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 Onchain team to perform an audit of their smart contract. The audit was conducted between 06/05/2024 and 09/0/2024.

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 Sepolia testnet at 0x32aA21f95D4dD93CB4398162032a021Bef08Be55.

Update. The Onchain team has responded to this report. The SHA-1 of the files are:
`599fc0cf6724de9812b2a763c206d4228a3d6014` token.sol

`aa56184851c204b1c5a31fdd686780e3218563f1` interfaces/IFactory.sol

`cc07efb4d01a249e3360fe4cf27cd2f2aa02e497` interfaces/IRouter.sol

`5d40bb46f54bd6d48a120faa37ddaf4f3c7654f9` libraries/SafeERC20.sol

2.1 Summary

Project name	Onchain
URL	https://onchaincoin.io/
Platform	Ethereum
Language	Solidity
Centralization level	High
Centralization risk	High

2.2 Contracts

Name	Address
OnchainToken	

3. Project centralization risks

The contract owner can update the whitelists of addresses that are exempt from transfer fees and limits. Initial fee value is 75/80% and initial transaction limit is 0.2%/2% (before and after the launch).

The owner can update transaction's fees, maximum amount and swap threshold within safety limits.

The owner can update list of trading pairs to apply transfer fees.

The owner can enable and disable the protection against MEV.

The owner can disable automatic mechanism of tax reducing and limit increasing, and then abstain from updating limits manually.

The owner can set treasury address, which receives swapped ETH, to a malicious one to impede or block transfers.

4. Found issues

Medium	3 (38%)
Low	2 (25%)
Info	3 (37%)

Cfe. OnchainToken

ID	Severity	Title	Status
CfeIb8	Medium	Possible DoS by gas	Resolved
CfeIb9	Medium	Unsafe cast	Resolved
CfeIb7	Medium	Lack of tests and documentation	Partially fixed
CfeIb5	Low	Lack of events	Resolved
CfeIb4	Low	Gas optimizations	Acknowledged
CfeIb6	Info	Swaps with 100% slippage	Acknowledged
CfeIc8	Info	The transfer delay check does not work if anti MEV functionality is disabled	Resolved
CfeIc9	Info	Account abstraction ERC4337 wallets not supported	Partially fixed

5. Contracts

Cfe. OnchainToken

Overview

An ERC-20 standard token with transfer fees individual for buys and sells to/from owner-controlled list of DEX pairs). Collected fees are swapped to native EVM coins once a threshold is reached, which are transferred out to an address controlled by the token owner. The token includes limit for transaction amount and MEV protection based on the last block for each holder and transaction signer.

Issues

Cfe1b8 Possible DoS by gas

Medium Resolved

The treasury address receives swapped ETH by `call()` method without specified gas amount. A malicious treasury can be used to perform gas spending actions or to significantly increase needed gas to complete token transfer.

```
    function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual override {
        if (!exemptFromFees[from] && !exemptFromFees[to]) {
            require(tradingAllowed, "Trading not active");
            amount -= handleTax(from, to, amount);
            checkLimits(from, to, amount);
        }

        super._transfer(from, to, amount);
    }

    function handleTax(
```

```
        address from,
        address to,
        uint256 amount
    ) internal returns (uint256) {
        if (balanceOf(address(this)) >= swapTokensAtAmt && !isPair[from]) {
            convertTaxes();
        }
        ...
    }

    function convertTaxes() private {
        ...
        if (ethBalance > 0) {
            (success, ) = treasury.call{value: ethBalance}("");
        }
    }
```

Recommendation

Limit the amount of forwarded gas.

Cfe1b9 Unsafe cast

Medium Resolved

The `updateTransactionLimit()` function include safety check to revert limit below 0.1% of total supply. However, unsafe cast uint256 to uint128 may result in bypassing this safety check and updating the `txLimits.transactionLimit` variable to unwanted value.

```
    function updateTransactionLimit(uint128 newNumInTokens) external onlyOwner {
        require(
            newNumInTokens >= ((totalSupply() * 1) / 1000) / (10 ** decimals()),
            "Too low"
        );
        txLimits.transactionLimit = uint128(
            newNumInTokens * (10 ** decimals())
        );
        emit UpdatedTransactionLimit(txLimits.transactionLimit);
    }
```

Recommendation

Use the SafeCast library from OpenZeppelin.

CfeIb7 Lack of tests and documentation

Medium Partially fixed

The project doesn't contain any tests and documentation. We recommend increasing test coverage, providing the documentation section, and adding documentation using NatSpec Format.

This would help in development, as well as simplify user interaction with the contract (including using the block explorer).

In particular, we found unclear the use of `tokensForTax` variable. The only way it may concern the workflow is following: if all transfer taxes are below 1e9, the accumulated fees wouldn't be swapped even after exceeding the `swapTokensAtAmt` threshold.

Also, the anti-MEV and transfer delay functionality is not described which makes it impossible to check whether the implemented functionality protects from attacks.

CfeIb5 Lack of events

Low Resolved

The functions `updateSwapTokensAmt()`, `enableTrading()`, `updateMevBlockerEnabled()`, `removeTransferDelay()` change important variable in the contract storage, but no event is emitted.

We recommend adding events to these functions to make it easier to track their changes offline.

CfeIb4 Gas optimizations

Low Acknowledged

- Unnecessary reads from storage in the constructor: `owner()` variable.

- Unnecessary stored data: the router contract doesn't hold any tokens and may not be included in the `exemptFromFees` list.
- Multiple reads from storage in the `swapTokensForETH()` function: `swapTokensAtAmt` variable.
- Redundant code: `if (contractBalance > 0)` check should be omitted in the `convertTaxes()` function.
- Unnecessary reads from storage in the `updateTransactionLimit()` function: `txLimits.transactionLimit` variable.
- Unnecessary reads from storage in the `setInternalTaxes()` function: after `phaseBlocks > 3600` the `dynamicTaxOn` switch should be disabled automatically.
- Redundant code: `tokensForTreasury` calculations in the `handleTax()` function: `tokensForTreasury = ... tax * taxes.totalTax / taxes.totalTax`

Cfe1b6 Swaps with 100% slippage

Info Acknowledged

DEX swapping in the `swapTokensForETH()` function is called with 100% slippage. A large `swapTokensAtAmt` amount may result in successful MEV actions, if the MEV protection is disabled.

Cfe1c8 The transfer delay check does not work if anti MEV functionality is disabled

Info Resolved

The function `checkLimits()` performs the transfer delay and anti-MEV checks. One of the checks for the transfer delay is that the token holder's last transfer was in previous blocks. However, the token holder's last transfer block is updated only if the anti-MEV functionality is enabled.

```
function checkLimits(address from, address to, uint256 amount) internal {
  ...
        if (transferDelayEnabled) {
            if (to != address(router) && to != address(lpPair)) {
                require(
                    _holderLastTransferBlock[tx.origin] < block.number,
                    "Transfer Delay"
```

```
                    );
                    require(
                        tx.origin == to,
                        "no buying to external wallets yet"
                    );
                }
            }
        if (antiMevEnabled) {
            if (isPair[to]) {
                require(
                    _holderLastTransferBlock[from] < block.number,
                    "Anti MEV"
                );
            } else {
                _holderLastTransferBlock[to] = block.number;
                _holderLastTransferBlock[tx.origin] = block.number;
            }
        }
    }
```

Recommendation

Update the holder's last transfer block if the transfer delay if case.

Cfelc9 Account abstraction ERC4337 wallets not supported

● Info Partially fixed

The contract uses `tx.origin` for the anti-MEV and transfer delay checks. These checks may incorrectly block token transfers on Account abstraction wallets. With ERC4337 the `tx.origin` is not the token holder who sends the transaction but the bundler which breaks the logic of the checks. Moreover, if transfer delay is enabled, account abstraction wallets won't be able to receive any tokens.

Update

The transfer delay functionality was removed from the token. But antiMEV checks can still incorrectly block some token transfers from Account abstraction wallets. If there is a token transfer transaction sent by a relayer/bunder, the subsequent tokens sells sent from this relayer/bundler that go into the same block will fail.

```
        if (antiMevEnabled) {
            if (isPair[to]) {
                require(
                    _holderLastTransferBlock[from] < block.number,
                    "Anti MEV"
                );
            } else {
                _holderLastTransferBlock[to] = block.number;
                _holderLastTransferBlock[tx.origin] = block.number;
            }
        }
```

6. Conclusion

3 medium, 2 low severity issues were found during the audit. 2 medium, 1 low issues were resolved in the update. The reviewed contracts are highly dependent on the owner's account. See the centralization risks chapter.

This audit includes recommendations on code improvement and the prevention of potential attacks.

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