soken

SMART CONTRACT SECURITY AUDIT

CZpegs

Scan and check this report was posted at Soken Github

June, 2022

Website: soken.io

Table of Contents

Disclaimer

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws. We took into consideration smart contract based algorithms, as well. Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it. Before making any judgments, you have to conduct your own independent research. We will discuss this in more depth in the following disclaimer - please read it fully.

DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Soken and its affiliates shall not be held responsible to you or anyone else, nor shall Soken provide any guarantee or representation to any person with regard to the accuracy or integrity of the report. Without any terms, warranties or other conditions other than as set forth in that exclusion and Soken excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills). The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Soken disclaims all responsibility and responsibilities and no claim against Soken is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent).

Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Our analysis contains following steps:

1. Project Analysis;

2. Manual analysis of smart contracts:
- Deploying smart contracts on any of the network(Ropsten/Rinkeby) using Remix IDE
- Hashes of all transaction will be recorded
- Behaviour of functions and gas consumption is noted, as well.

3. Unit Testing:
- Smart contract functions will be unit tested on multiple parameters and under multiple conditions to ensure that all paths of functions are functioning as intended.
- In this phase intended behaviour of smart contract is verified.
- In this phase, we would also ensure that smart contract functions are not consuming unnecessary gas.
- Gas limits of functions will be verified in this stage.

4. Automated Testing:
- Mythril
- Oyente
- Manticore
- Solgraph

Terminology

We categorize the finding into 4 categories based on their vulnerability:

- Low-severity issue — less important, must be analyzed
- Medium-severity issue — important, needs to be analyzed and fixed
- High-severity issue —important, might cause vulnerabilities, must be analyzed and fixed
- Critical-severity issue —serious bug causes, must be analyzed and fixed.

Limitations

The security audit of Smart Contract cannot cover all vulnerabilities. Even if no vulnerabilities are detected in the audit, there is no guarantee that future smart contracts are safe. Smart contracts are in most cases safeguarded against specific sorts of attacks. In order to find as many flaws as possible, we carried out a comprehensive smart contract audit. Audit is a document that is not legally binding and guarantees nothing.

Audit Details

Project Name: **CZpegs**

Language: **Solidity**

Social Profiles

Project Website: **https://www.czpegs.com/**

Project GitHub: **https://github.com/czpegs**

Project Telegram: **https://t.me/CZpegs**

Project Twitter: **https://twitter.com/CZpegs**

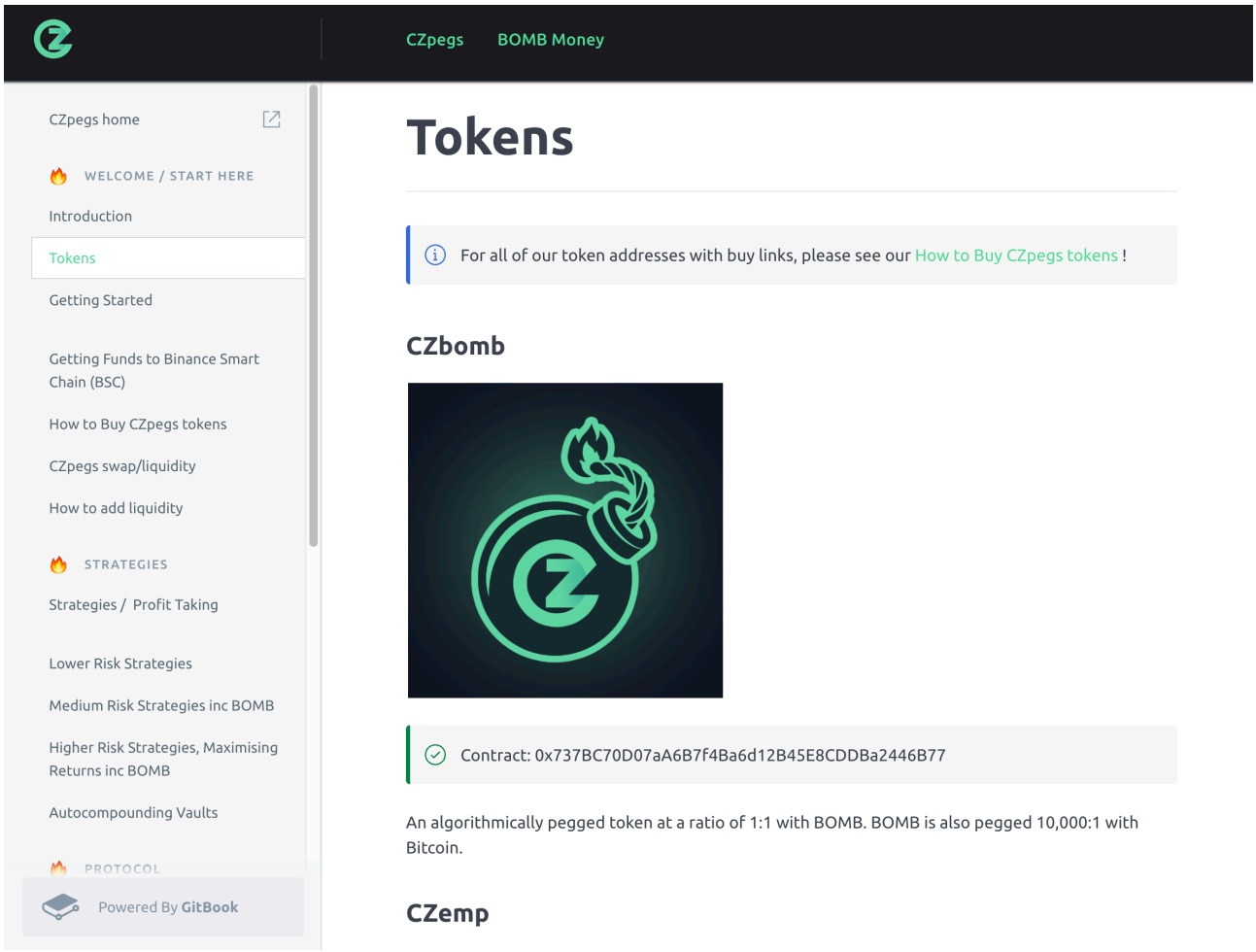
Project Discord: **https://discord.com/invite/k3aZEu25wE**

About the project

“Czpegs is based upon the work of an accumulation of TOMB finance forks over the last few months. The project is run and lead by the doxxed team at BOMB Money, lead by CEO Aaron Shames! We have taken the best parts from all of the best forks and combined them into one polished, final product! These extra additions will serve as a foundation for BOMB and EMP, providing constant buy pressure as well as offering high yield to BNB enthusiasts all around Defi! As far as networks go, Binance Smart Chain (BSC) seemed like the perfect choice again with its fast transaction speeds, low gas fees, and large user base. Plus, it's the home of BOMB and EMP!"

Docs Review

CZpegs Docs, tokenomics and roadmap have been reviewed by Soken



Docs: **https://docs.czpegs.com/**

Protocol Contract Addresses

CZbomb treasury

0xFF738aDa86dBdD2A55c3bD9f2019b8e5755BFBD0

CZbnb treasury

0xb9B0a97689068ee4d81A4e2609480EB2F2A4f37a

CZemp treasury

0xf8aD4F4738F21873fA2F7f79866b631dea020819

CZbusd treasury

0xE75E9c07ACC55E4D521ADD4284eD269B9961aE6B

CZbomb boardroom

0xF0cE68bfE5B9B8470C99fE66E4bAfa31e178C0f0

CZbnb boardroom

0xBb4a1a464998C0E3c1Fa296F8274539017cb3B21

Link: **https://docs.czpegs.com/protocol/other-protocol-contract-addresses**

Vulnerabilities checking

Issue Description	Checking Status
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Timestamp Dependence	Completed
Shadowing State Variables	Completed
DoS with Failed Call	Completed
DoS with Block Gas Limit	Completed
Outdated Complier Version	Completed
Assert Violation	Completed
Use of Deprecated Solidity Functions	Completed
Integer Overflow and Underflow	Low-issues
Function Default Visibility	Completed
Malicious Event Log	Completed
Math Accuracy	Completed
Design Logic	Completed
Fallback Function Security	Completed
Cross-function Race Conditions	Completed
Safe Zeppelin Module	Completed

Security Issues

1) Integer Overflow/Underflow : Medium-severity

Oracle.sol - Line: 68 - 87

OracleEMP.sol - Line: 68 - 87

OracleBUSD.sol - Line: 68 - 87

OracleBNB.sol - Line: 68 - 87

OracleBOMB.sol - Line: 68 - 87

```
68    function update() external checkEpoch {
69        (uint256 price0Cumulative, uint256 price1Cumulative, uint32 blockTimestamp) = UniswapV2OracleLibrary.currentCumulativePrices(address(pair));
70        uint32 timeElapsed = blockTimestamp - blockTimestampLast; // overflow is desired
71
72        if (timeElapsed == 0) {
73            // prevent divided by zero
74            return;
75        }
76
77        // overflow is desired, casting never truncates
78        // cumulative price is in (uq112x112 price * seconds) units so we simply wrap it after division by time elapsed
79        price0Average = FixedPoint.uq112x112(uint224((price0Cumulative - price0CumulativeLast) / timeElapsed));
80        price1Average = FixedPoint.uq112x112(uint224((price1Cumulative - price1CumulativeLast) / timeElapsed));
81
82        price0CumulativeLast = price0Cumulative;
83        price1CumulativeLast = price1Cumulative;
84        blockTimestampLast = blockTimestamp;
85
86        emit Updated(price0Cumulative, price1Cumulative);
87    }
```

An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum storage of a variable type. Integers overflow or underflow may prove fatal when during an arithmetic operation, the number goes over or under the designated limit. This may prove fatal during calculations related to ether or tokens.

Recommendation:

Solidity compiler **versions >=0.8.0** automatically handle overflow and underflow validations. If you're using a lower solidity version, it is recommended to use the **SafeMath** library to protect the arithmetic operations.

2) Loop consuming excessive gas : Low-severity

Distributor.sol - Line: 15-17

Treasury.sol - Line: 450-452

TreasuryBOMB.sol - Line: 449-451

TreasuryEMP.sol - Line: 449-451

TreasuryBUSD.sol - Line: 449-451

TreasuryBNB.sol - Line: 449-451

```
15            for (uint256 i = 0; i < distributors.length; i++) {
16                distributors[i].distribute();
17            }
```

```
450           for (uint8 entryId = 0; entryId < excludedFromTotalSupply.length; ++entryId) {
451               balanceExcluded = balanceExcluded.add(tokenErc20.balanceOf(excludedFromTotalSupply[entryId]));
452           }
```

Recommendation:

Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit, which can cause the complete contract to be stalled at a certain point. Therefore, loops with a bigger or unknown number of steps should always be avoided.

Resolution:

The team acknowledged this issue and decided not to change the current codebase.

3) UniswapV2Library Contains Network Dependent Code

Informational

UniswapV2Library.sol - Line: 32

```
31                keccak256(abi.encodePacked(token0, token1)),
32                hex"96e8ac4277198ff8b6f785478aa9a39f403cb768dd02cbee326c3e7da348845f" // init code hash
```

The init code hash used in the pairFor() function is specific to the Uniswap on Ethereum mainnet and will not correctly calculate pairs when deployed on other networks.

Recommendation:

Update the library to be compatible with the DEX and chain it will be deployed to.

Resolution:

Team comment: “Issue 4 is irrelevant since we don't need to calculate pair addresses"

Conclusion

Smart contracts are free from any critical or high-severity issues.

NOTE: Please check the disclaimer above and note, that audit makes no statements or warranties on business model, investment attractiveness or code sustainability.

Soken Contact Info

Website: www.soken.io

Mob: (+1)416-875-4174

32 Britain Street, Toronto, Ontario, Canada

Telegram: @team_soken

GitHub: sokenteam

Twitter: @soken_team