Jorge Rodriguez

CYBERSEC CONTRACT AUDIT REPORT
BASIX PROTOCOL - CTDSEC.com

Introduction

During February of 2021, BASIX PROTOCOL engaged CTDSec to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. Basix Protocol provided CTDSec with access to their code repository and whitepaper.

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bugfree status. The audit documentation is for discussion purposes only.

I always recommend having a bug bounty program opened to detect future bugs.

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of the Basix Protocol contract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:

- BasixProtocol.sol - MD5: 9D9C88C169EA67A6D104A9F5BCBDD65A
- BasixToken.sol - MD5: FE12286DE812FFF4277E3DB3F8878A1A
- Deployer.sol - MD5: 2CD9F9DA74C4A7FC6836D03836862A01
- Interfaces.sol - MD5: B2D656530E821EA62FD7E3906ABF603F
- Migrations.sol - MD5: E72540E10B19090593F102AF4E0CBEF6
- Oracle.sol - MD5: 3900E44BECB2972B6721FC4668AE3B48
- Orchestrator.sol - MD5: 11173136BA591DC70B4ABECE077C44EE
- Pool.sol - MD5: 7C84496AED8E55A707C6B3CAA5748DC1
- BasixTeamLock.sol - MD5: 5E638C731CA02BF416CC03067C1C6497

Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

Correctness of the protocol implementation [Result OK]

User funds are secure on the blockchain and cannot be transferred without user permission [Result OK]

Vulnerabilities within each component as well as secure interaction between the network components [Result OK]

Correctly passing requests to the network core [Result OK]

Data privacy, data leaking, and information integrity [Result OK]

Susceptible to reentrancy attack [Result OK]

Key management implementation: secure private key storage and proper management of encryption and signing keys [Result OK]

Handling large volumes of network traffic [Result OK]

Resistance to DDoS and similar attacks [Result OK]

Aligning incentives with the rest of the network [Result OK]

Any attack that impacts funds, such as draining or manipulating of funds [Result OK]

Mismanagement of funds via transactions [SOLVED BY DEV TEAM]

Inappropriate permissions and excess authority [Result OK]

Special token issuance model [Result OK]

Vulnerabilities

ISSUES

HIGH

Funds locked forever.

Due to a code error, the team's funds would be locked forever.

Location:

```
function withdraw() external {
    require(block.timestamp == releaseTime(), "BasixTeamLock: tokens are still locked");
    uint256 amount = basix().balanceOf(address(this));
    require(amount > 0, "BasixTeamLock: tokens already released");
    basix().transfer(beneficiary(), amount);
}
```

Team update: The development team fixes the function so that the funds can be unlocked after one year.

```
function withdraw() external {
    require(block.timestamp >= releaseTime(), "BasixTeamLock: tokens are still locked");
    uint256 amount = basix().balanceOf(address(this));
    require(amount > 0, "BasixTeamLock: tokens already released");
    basix().transfer(beneficiary(), amount);
}
```

MEDIUM

Multiplication after division.

Such as other languages from example Javascript that have floating point errors solidity instead have rounding errors. If the division is done before the multiplication, the rounding errors can increase dramatically.By doing all our multiplications first, we mitigate rounding related issues as much as possible. The computations can be much more complex and forming them into a multiplication first formula can be challenging at times.

Location:

```
uint256 grainValue = value.mul(_grainsPerFragment);
uint256 grainFee = grainValue.div(10000).mul(TRANSFER_FEE);
uint256 newGrainsValue = grainValue - grainFee;
uint256 newValue = newGrainsValue.div(_grainsPerFragment);
```

Team update: As they are working with huge values, the reversal cannot be done as it could produce an overflow.

Contract transparency

After the deploying team will renounce ownership, the contract will be 100% decentralized.

Team locked liquidity via unicrypt and team tokens are locked for 1 year.

Every TX burn supply.

Architecture

Since the images are too large and cannot be attached to the document due to loss of quality, we attach hyperlinks to be able to view them correctly.

Basix Protocol Architecture: https://pasteboard.co/JOcoohu.png

Summary of the Audit

The contract is safe and correctly applied according to token economics.

After reviewing the contract we came to the conclusion that is safe to deploy.