Cygnus Finance Audit

Introduction

This audit report evaluates the overall security of the **Cygnus Finance** smart contracts. The primary goal is to ensure the reliability and robustness of these contracts through a comprehensive assessment of the system architecture and codebase. By identifying potential vulnerabilities and weaknesses, this audit aims to enhance the security posture of Cygnus Finance, fostering trust and confidence among its users and stakeholders.

Disclaimer

This audit does not provide any guarantees regarding the security of the code. Security is a complex and multifaceted challenge, and a single audit cannot uncover all potential issues. We strongly recommend conducting multiple independent audits and implementing a public bug bounty program to enhance the security of the code.

Audit Details

- **Author:** Daniil Sedov
- **Date:** 7th August 2024
- **Commit:** `4330cbfb596c6f03a26d72dc1be5b16fdfac8ad2`
- **Language:** Tact
- **Methods:** Manual review, static analysis

Scope of Audit

The focus of this audit was to verify whether the smart contracts are secure, resilient, and functioning correctly according to the specifications. The audit included a review of the overall system architecture and logic, as well as a thorough examination of the contract source code. The code review encompassed checks for correctness, optimization, and readability.

Files included in the scope:

- `cgUSDt.tact`
- `Jetton.tact`

- `MessagesJetton.tact`
- `MessagesNFT.tact`
- `Minter.tact`
- `mockUSDt.tact`
- `NFT.tact`

Overview

Overall, the system seems to be implemented correctly, and no critical problems were found. However, there are considerations about gas constants calculation and multiple minor code quality issues.

Findings

The audit uncovered a total of **9** issues, categorized as follows:

- **Critical:** Issues that could cause system-wide failures, theft, freezing, or loss of users' funds by a third party. These must be addressed before release.
- **Major:** Issues that could cause the failure of some components, freezing or loss of funds for users, and create delays and lags in the system. These must be addressed before release.
- **Medium:** Issues that could potentially cause significant problems but are more challenging to exploit or encounter under typical conditions. These must be addressed before release.
- **Minor:** Small bugs that don't cause significant problems, inefficient code, and poor practices. These are recommended to be addressed before release.
- **Informational:** Issues related to code readability, naming, and other non-operational problems. These are recommended to be addressed before release.

Details for each of these categories are explained below.

Critical

(No critical issues found)

Major

(No major issues found)

Medium

MED-1

Location

`Jetton.tact:28`, `Jetton.tact:113-114`, `NFT.tact:22-23`, `NFT.tact:179-180`

```
const gas_consumption: Int = ton("0.01");
```

Description

Fees-related constants are hardcoded as Toncoin values, which may cause problems if gas and storage prices ever change in TON, which is possible (and has already happened before). In case of fees increase, the contract will just break, and in case of fees decrease, it will be excessively expensive for users.

Recommendation

It is strongly recommended to use native TVM instructions such as `GETGASFEE` to calculate fee values, the usage of which you can find in some of the modern smart contract implementations, like this one.

Minor

MIN-1

Location

`mockUSDt.tact:18`

```
let ctx: Context = context();
```

Description

The `ctx` variable is assigned but never used.

Recommendation

Remove the unused variable.

MIN-2

Location

`NFT.tact:60`

```
  let zero: Int = s.loadUint(192);
```

Description

The `zero` variable is assigned but never used.

Recommendation

Remove the unused variable.

MIN-3

Location

`cgUSDt.tact:22`, `mockUSDt.tact:19`

```
  require(self.mintable, "Not mintable");
```

Description

This check is not needed here because the sender address is being checked in the `self.mint` method which is called later.

Recommendation

Remove the line with the unneeded check.

MIN-4

Location

`NFT.tact:59-61`

```
let s: Slice = beginCell().storeUint(msg.item_index,256).asSlice();
let zero: Int = s.loadUint(192);
let index: Int = s.loadUint(64);
```

Description

This is not optimal at all and consumes over 700 gas just to calculate these two values.

Recommendation

Use binary shift operators to calculate these two values instead of cell composing and parsing.

MIN-5

Location

`NFT.tact:117-118`

```
let s: Slice =
beginCell().storeUint(0,192).storeUint(src.query_id,64).asSlice();
let index: Int = s.loadUint(256);
```

Description

This is not optimal at all and consumes over 600 gas just to calculate this one value.

Recommendation

Use binary shift operators to calculate this value instead of cell composing and parsing.

Informational

INF-1

Location

`Jetton.tact:238`, `Jetton.tact:208`, `Jetton.tact:176`, `Jetton.tact:139`, `Jetton.tact:89`, `Jetton.tact:15`, `NFT.tact:91`, `NFT.tact:112`, `NFT.tact:132`, `NFT.tact:213`, `NFT.tact:231`, `NFT.tact:325`

```
msg_value = msg_value - (storage_fee + self.gas_consumption);
// other locations are similar
```

Description

The `A = A - B` pattern is used instead of the augmented assignment operator `-=`.

Recommendation

Use the `-=` operator.

INF-2

Location

`Jetton.tact:243`, `Jetton.tact:247`, `Jetton.tact:170`, `Jetton.tact:57`

```
self.balance = self.balance + src.amount;
// other locations are similar
```

Description

The `A = A + B` pattern is used instead of the augmented assignment operator `+=`.

Recommendation

Use the `+=` operator.

INF-3

Location

`NFT.tact:83-35`

```
let current_item_index: Int = self.next_item_index;
let nft_init: StateInit = self.get_nft_item_init(current_item_index);
self.next_item_index = current_item_index + 1;
```

Description

The `current_item_index` variable is not really needed here.

Recommendation

Rewrite as below.

```
let nft_init: StateInit = self.get_nft_item_init(self.next_item_index);
self.next_item_index += 1;
```