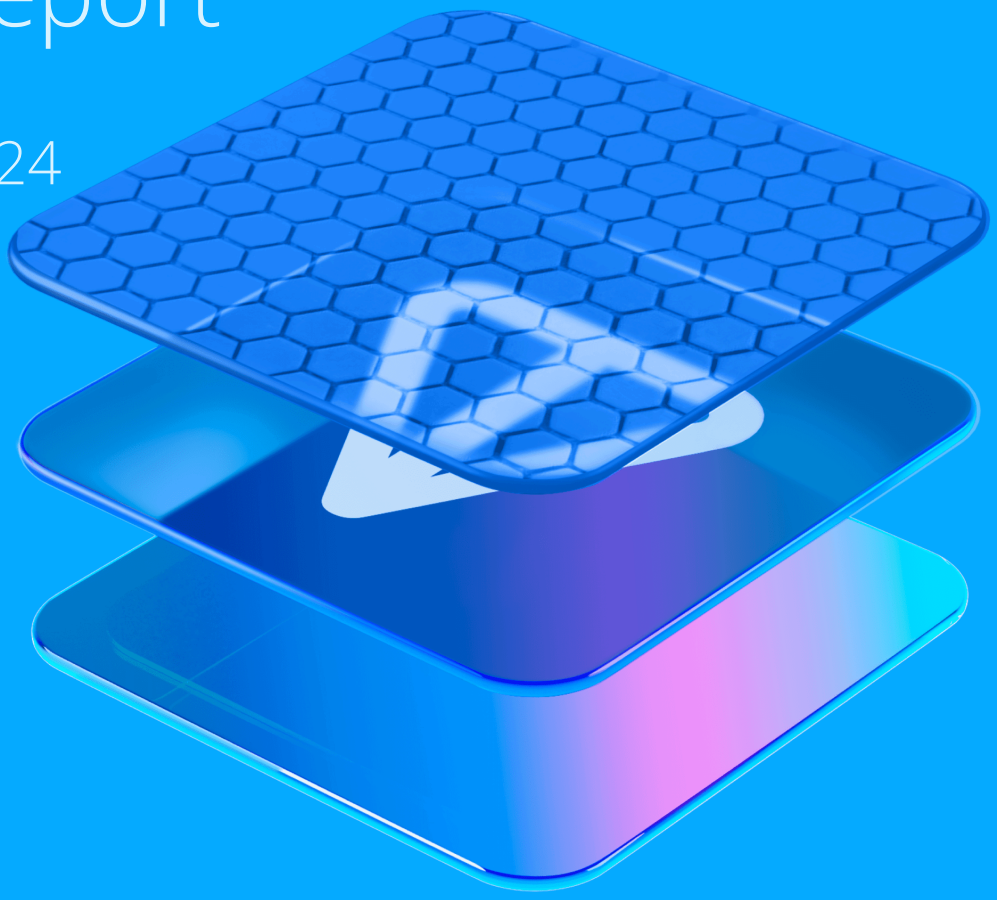
Miniton Smart Contract Audit Report

Fri Jul 19 2024



contact@bitslab.xyz https://twitter.com/tonbit_

TonBit

Miniton Smart Contract Audit Report

1 Executive Summary

1.1 Project Information

Description	MiniTon is a TON-backed crypto social competitive platform where players can compete internationally for real prizes and meaningful connections.
Type	Game
Auditors	TonBit
Timeline	Tue Jul 09 2024 - Fri Jul 19 2024
Languages	Tact
Platform	Ton
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/MultiversePlay/miniton-sc/
Commits	06b8444ca1e896a4763770d65dd9feba4b6592b4 c4a3db0e83ff7b887bb134c5f4cf19d830fb92d3 9ee21ae100684bbe26f1645d534e9298084ca4af

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
MTMC	sources/MiniTonMatchContract.tact	c62efe7cbb4e8d24dd1f5e4c47b9d0ccca15c80d

1.3 Issue Statistic

Item	Count	Fixed	Partially Fixed	Acknowledged
Total	6	4	1	1
Informational	1	1	0	0
Minor	3	2	0	1
Medium	2	1	1	0
Major	0	0	0	0
Critical	0	0	0	0

1.4 TonBit Audit Breakdown

TonBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by MVP to identify any potential issues and vulnerabilities in the source code of the Miniton smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 6 issues of varying severity, listed below.

ID	Title	Severity	Status
MTM-1	Continuing to Execute Trades when Commission Collection Fails	Medium	Fixed
MTM-2	Centralization Risk	Medium	Partially Fixed
MTM-3	Failure to Judge Gas Charges in Advance	Minor	Acknowledged
MTM-4	Lack of Events Emit	Minor	Fixed
MTM-5	Meaningless Virable	Minor	Fixed
MTM-6	Code Optimisation	Informational	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Miniton Smart Contract :

Owner

- The Owner can record match info through `MatchInfoMsg` message.
- The Owner can initiate a withdraw request through `MasterActionRequestMsg` message.
- The Owner can send the prize to the winner through `SendPrizeMsg` message.
- The Owner can remove the match dumps through `RemoveMatchDumpMsg` message.
- The Owner can set the minimum balance of the contract through `MinBalanceMsg` message.

Voter

- The Voter can vote a withdraw request through `VoteMsg` message.

4 Findings

MTM-1 Continuing to Execute Trades when Commission Collection Fails

Severity: Medium

Status: Fixed

Code Location:

sources/MiniTonMatchContract.tact#163

Descriptions:

In `sendMatchCommission()`, when the balance after the commission is collected is less than the specified value, the collection of the commission will be skipped and will not affect the execution of the subsequent logic. Under certain circumstances, it may be possible to skip the collection of the congestion fee and execute it successfully. This can result in loss of funds and is difficult to trace.

Suggestion:

It is recommended to ensure you are aware of this or returns an error when skipping commission collection.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

MTM-2 Centralization Risk

Severity: Medium

Status: Partially Fixed

Code Location:

sources/MiniTonMatchContract.tact

Descriptions:

Centralization risk was identified in the smart contract. The contract owner can withdraw all the assets in the contract.

Suggestion:

It's recommended that measures be taken to reduce the centralization issue like using multi-sig.

Resolution:

The client partly fixed the issue by using a voting mechanism to withdraw funds from the contract.

MTM-3 Failure to Judge Gas Charges in Advance

Severity: Minor

Status: Acknowledged

Code Location:

sources/MiniTonMatchContract.tact

Descriptions:

Gas consumption is not predicted in advance, and when there is not enough gas for a transaction it will consume gas and cause the transaction to fail.

Suggestion:

It is recommended to test the gas consumption in advance and judge the gas consumption in advance to decide whether to execute the subsequent logic or the administrator should use enough gas in the invocation.

Resolution:

The client responded that they will calculate the gas on the server side and check if the contract balance is sufficient.

MTM-4 Lack of Events Emit

Severity: Minor

Status: Fixed

Code Location:

sources/MiniTonMatchContract.tact#261,278,292

Descriptions:

The contract lacks appropriate events for monitoring sensitive operations, which could make it difficult to track sensitive actions or detect potential issues. For example, when receiving message `WithdrawMsg` , `Withdraw all` and `MinBalanceMsg` .

Suggestion:

It is recommended to emit events for those important functions.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

MTM-5 Meaningless Virable

Severity: Minor

Status: Fixed

Code Location:

sources/MiniTonMatchContract.tact#218

Descriptions:

The variable `request.action` is not modified in the contract, so using this variable in the VoteMsg function seems to be meaningless.

Suggestion:

It is recommended to confirm whether the usage of this variable aligns with your design.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

MTM-6 Code Optimisation

Severity: Informational

Status: Fixed

Code Location:

sources/MiniTonMatchContract.tact#114,218,251

Descriptions:

When making if judgements, consider simplifying the representation of truth values to improve code simplicity and efficiency. For example:

```
if (exist == false) {
        self.matchCount += 1;
    }
```

Suggestion:

It is recommended to optimise the above code as

```
if (!exist) {
        self.matchCount += 1;
    }
```

Resolution:

This issue has been fixed. The client has adopted our suggestions.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.