# Ton Market

# Security Audit Report

May 29, 2025

# 1. INTRODUCTION

## 1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

## 1.2 Security Assessment Methodology

BugBlow utilized a widely adopted approach to performing a security audit. Below is a breakout of how our team was able to identify and exploit vulnerabilities.

### 1.2.1 Code and architecture review:

- ➢ Review the source code.
- ➢ Read the project's documentation.
- ➢ Review the project’s architecture.

Stage goals

- ➢ Build a good understanding of how the application works.

### 1.2.2 Check code against vulnerability checklist:

- ➢ Understand the project’s critical assets, resources, and security requirements.
- ➢ Manual check for vulnerabilities based on the auditor's checklist.
- ➢ Run static analyzers.

Stage goals

- Identify and eliminate typical vulnerabilities (gas limit, replay, flash-loans attack, etc.)

## 1.2.3 Threat & Attack Simulation:

- Analyze the project's critical assets, resources and security requirements.
- Exploit the found weaknesses in a safe local environment.
- Document the performed work.

Stage goals

- Identify vulnerabilities that are not listed in static analyzers that would likely be exploited by hackers.
- Develop Proof of Concept (PoC).

## 1.2.4 Report found vulnerabilities:

- Discuss the found issues with developers
- Show remediations.
- Write an initial audit report.

Stage goals

- Verify that all found issues are relevant.

## 1.2.5 Fix bugs & re-audit code:

- Help developers fix bugs.
- Re-audit the updated code again.

Stage goals

- Double-check that the found vulnerabilities or bugs were fixed and were fixed correctly.

## 1.2.4. Deliver final report:

- The Client deploys the re-audited version of the code
- The final report is issued for the Client.

Stage goals

- Provide the Client with the final report that serves as security proof.

## Severity classification

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

| Severity | Description |
|---|---|
| Critical | Vulnerabilities leading to assets theft, fund access locking, or any other loss of funds. |
| High | Vulnerabilities that can trigger contract failure. Further recovery is possible only by manual modification of the contract state or replacement. |
| Medium | Vulnerabilities that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds. |
| Low | Vulnerabilities that do not have a significant immediate impact and could be easily fixed. |

# 1.3 Project Overview

Tonmarket is an on-chain prediction market for the TON blockchain, where users can buy and sell “yes” or “no” shares on the outcome of real-world events using a bonding curve AMM. After the event concludes, an oracle resolves the market, allowing holders of the winning shares to claim their share of the pooled funds. The smart contracts are structured with clear access controls, automated fee handling, and protection against common security issues like overflows or unauthorized withdrawals. With its transparent, self-contained logic and flexible market parameters, this project provides an interesting and modern approach to decentralized event forecasting on TON.

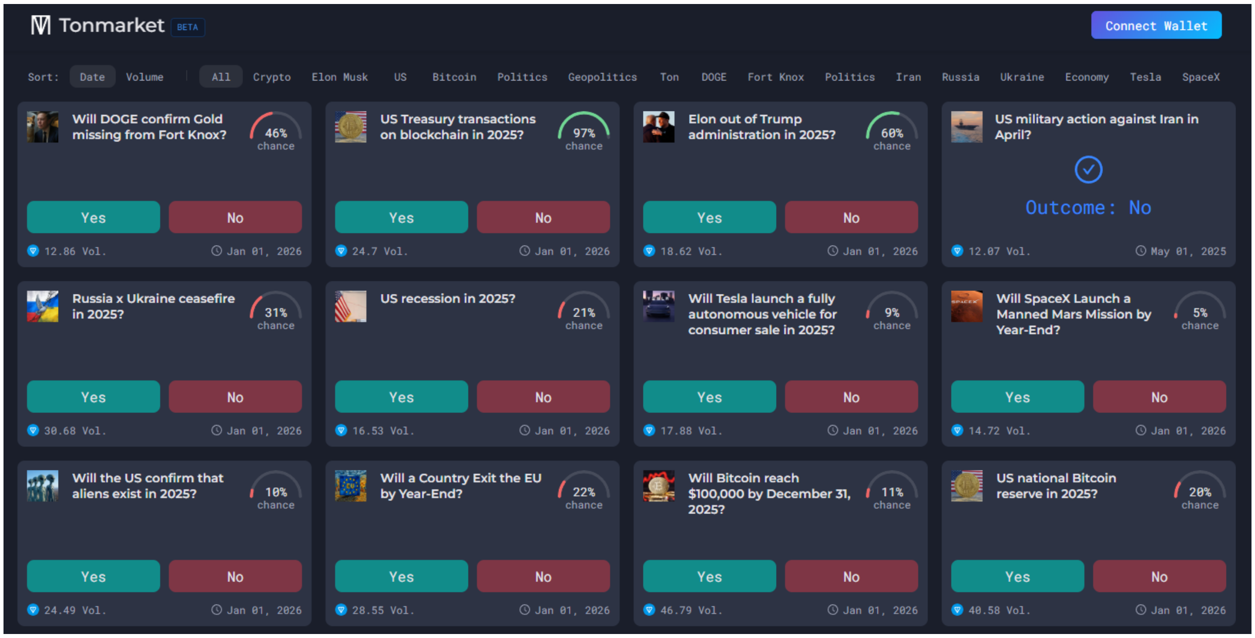
Tonmarket BETA
Connect Wallet
Sort: Date Volume
All Crypto Elon Musk US Bitcoin Politics Geopolitics Ton DOGE Fort Knox Politics Iran Russia Ukraine Economy Tesla SpaceX
Will DOGE confirm Gold missing from Fort Knox? 46% chance
Yes No
12.86 Vol. Jan 01, 2026
US Treasury transactions on blockchain in 2025? 97% chance
Yes No
24.7 Vol. Jan 01, 2026
Elon out of Trump administration in 2025? 60% chance
Yes No
18.62 Vol. Jan 01, 2026
US military action against Iran in April?
Outcome: No
12.07 Vol. May 01, 2025
Russia x Ukraine ceasefire in 2025? 31% chance
Yes No
30.68 Vol. Jan 01, 2026
US recession in 2025? 21% chance
Yes No
16.53 Vol. Jan 01, 2026
Will Tesla launch a fully autonomous vehicle for consumer sale in 2025? 9% chance
Yes No
17.88 Vol. Jan 01, 2026
Will SpaceX Launch a Manned Mars Mission by Year-End? 5% chance
Yes No
14.72 Vol. Jan 01, 2026
Will the US confirm that aliens exist in 2025? 10% chance
Yes No
24.49 Vol. Jan 01, 2026
Will a Country Exit the EU by Year-End? 22% chance
Yes No
28.55 Vol. Jan 01, 2026
Will Bitcoin reach $100,000 by December 31, 2025? 11% chance
Yes No
46.79 Vol. Jan 01, 2026
US national Bitcoin reserve in 2025? 20% chance
Yes No
40.58 Vol. Jan 01, 2026

Figure 1. Ton Market Overview.

## 1.4 Project Dashboard

### Project Summary

| Title | Ton Market Security Audit |
|---|---|
| Client name | Ton Market Labs |
| Project name | Ton Market |
| Timeline | 16.05.2025 - 23.05.2025 |
| Number of Auditors | 3 |

### Project Last Log

| Date | Commit Hash | Note |
|---|---|---|
| 26.03.2025 | 3f40029eb32de621b2899b890da05548219aa5b4 | update readme |

### Project Scope

The audit covered the following smart contract files.

| File name |
|---|
| contracts/buy.fc |
| contracts/common.fc |
| contracts/owner.fc |
| contracts/redeem.fc |
| contracts/resolve.fc |
| contracts/sell.fc |
| contracts/ton_market.fc |
| scripts/deployTonMarket.ts |
| scripts/incrementTonMarket.ts |

## 1.5 Summary of findings

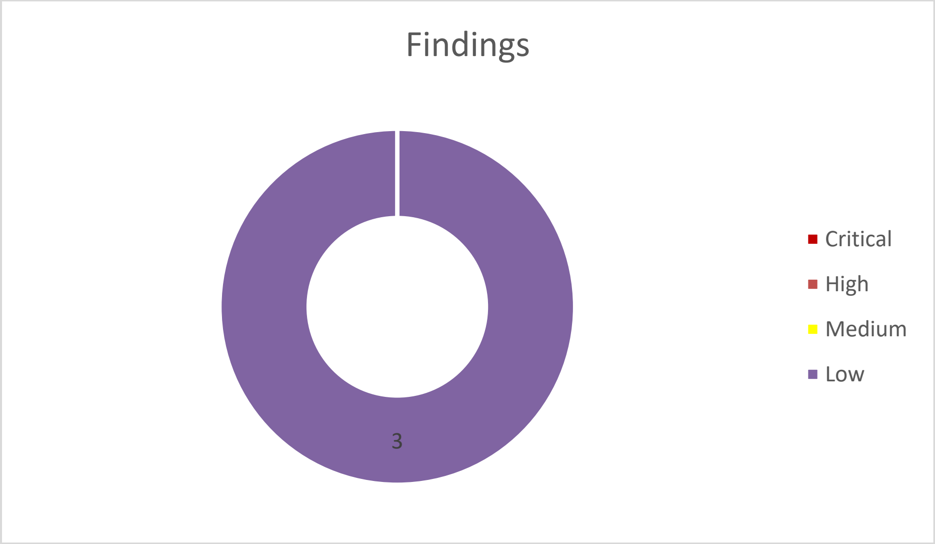

Figure 1. Findings chart

| Severity | # of Findings |
|---|---|
| CRITICAL | 0 |
| HIGH | 0 |
| MEDIUM | 0 |
| LOW | 3 |

# 2. FINDINGS REPORT

## 2.1 Critical

Not found

## 2.2 High

Not found

## 2.3 Medium

Not found

## 2.4 Low

### 2.4.1 Bonding Curve Slight Overcharge on Buy

Status: Fixed

Description

The bonding curve price calculation in buy.fc slightly overcharges users on multi-share purchases due to a formula bug (N*N/2 instead of the mathematically correct N*(N-1)/2). This is a fairness issue, not a direct security risk, but it makes large purchases less efficient for users.

#### Remediation

If you want the bonding curve math to match standard AMM design:

```
int actual_net_amount = INITIAL_PRICE * shares_to_buy + PRICE_SLOPE * total_shares * shares_to_buy +
(PRICE_SLOPE * shares_to_buy * (shares_to_buy - 1)) / 2;
```

### 2.4.2 No challenge period

#### Status: Acknowledged

#### Description

The current design resolves the market outcome as soon as the oracle submits its result, enabling immediate user payouts without delays. While this approach provides fast and seamless settlement, it does not include a dispute or challenge period before the outcome is finalized. This makes the user experience very responsive and simple, but also means users rely on the oracle’s accuracy and integrity for each market. For projects or applications seeking additional transparency and community involvement we recommend adding implement it in the future. Even if it is just waiting for the payout, the users would have time to provide their input on the outcome.

#### Remediation

Implement challenge periods in the future.

### 2.4.3 Oracle control

#### Status: Acknowledged

#### Description

Although not critical now, the oracle responsible for resolving market outcomes is secured by a single private key. This design keeps operations straightforward but creates a central point of trust: if that key is ever lost or compromised, the market could be resolved incorrectly or unfairly. For stronger security and reliability, it is recommended to upgrade to a multi-signature (multisig) approach, where multiple independent parties must approve any outcome, or to use secure multi-party computation (MPC). With MPC, the key is split into several shares and kept on different servers or with different individuals, only coming together in memory to authorize a resolution. Both approaches greatly reduce risk, making the oracle far more resilient to loss or compromise and increasing user confidence in the market's integrity.

### Remediation

Replace the single oracle key with either a multi-signature (multisig) setup or an MPC (multi-party computation) scheme, distributing key control across multiple parties or locations to ensure that no single individual can resolve the market alone.

# CONCLUSION

This security audit found the TON Market smart contracts to be well-designed and secure, with no critical, high, or medium severity issues detected. Only minor issues were identified, such as missed fee collection on sell transactions and a slight overcharge in the bonding curve formula—both of which have already been fixed. Other low-severity items, including the lack of a challenge period and centralized oracle control, are noted for future improvement but do not pose immediate risks to user funds or market integrity. Overall, the project demonstrates strong security and sound development practices, offering users a safe and reliable prediction market platform.

# REFERENCES

[1] Dispute Periods in Oracles - https://reality.eth.link/docs/html/