Wakehacker Report

Contract address	0x88df592f8eb5d7bd38bfef7deb0fbc02cf3778a0
Chain ID	1
Report version	1.0
Last edit	November 17, 2025

"Static analysis or stay rekt"

Document Revisions

1.0	Wakehacker Report - 0x88df592f8eb5d7bd38bfef7deb0fbc02cf3778a0 on chain 1	17.11.2025

Contents

Overview

This report was generated using Wakehacker, an automated vulnerability analysis tool. Wakehacker utilizes Wake with additional detectors to perform comprehensive AI and static analysis.

To identify potential vulnerabilities and issues in smart contracts Wake framework utilizes:

- Code structure and patterns
- Control flow graph
- Data flow graph
- Common vulnerability patterns
- Contract interactions

The findings presented in this report are based on automated analysis optimized for precision, aiming for a low false-positive rate. The detection is not optimized for recall—it doesn't target finding all issues (which come at the cost of a high false-positive rate). This code review should be complemented with additional manual code review for a complete security assessment.

Disclaimer

The best effort has been put into finding known vulnerabilities in the system, however automated findings shouldn't be considered as a complete list of all existing issues. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Finding Classification

Each finding is classified by two independent ratings: *Impact* and *Confidence*.

Impact

Measuring the potential consequences of the issue on the system.

- **High** - Code that activates the issue will lead to undefined or catastrophic consequences for the system.
- **Medium** - Code that activates the issue will result in consequences of serious substance.
- **Low** - Code that activates the issue will have outcomes on the system that are either recoverable or don't jeopardize its regular functioning.
- **Warning** - The issue represents a potential security concern in the code structure or logic that could become problematic with code modifications.
- **Info** - The issue relates to code quality practices that may affect security. Examples include insufficient logging for critical operations or inconsistent error handling patterns.

Confidence

Indicating the probability that the identified issue is a valid security concern.

- **High** - The analysis has identified a pattern that strongly indicates the presence of the issue.
- **Medium** - Evidence suggests the issue exists, but manual verification is recommended.
- **Low** - Potential indicators of the issue have been detected, but there is a significant possibility of false positives.

Executive Summary

Revision 1.0

The audit of the Tellor decentralized oracle protocol at address 0x88df592f8eb5d7bd38bfef7deb0fbc02cf3778a0 on Ethereum mainnet identified a total of 2 detections, all stemming from a single detector type. Both detections were flagged with high confidence levels, though notably no high impact or critical severity issues were discovered. The findings suggest a relatively secure implementation of the protocol's smart contract infrastructure, with the identified issues being of lower severity nature that warrant attention but do not pose immediate critical risks to the protocol's core functionality or user funds.

Findings

Summary by Impact

0	0	2	0	0	2
High	Medium	Low	Warning	Info	Total

Complete List

Finding Title	Impact	Reported	Status
L1: Missing event emission for deity role change (changeDeity)	Low	1.0	Reported
L2: Missing event emission for owner change (changeOwner)	Low	1.0	Reported

L1: Missing event emission for deity role change (changeDeity)

Impact	Low	Confidence	High
Target	contracts/TellorMaster.sol	Detection	Wake AI

Description

The contract allows the privileged deity address (an upgrade authority) to be changed via `changeDeity` without emitting any event. This eliminates an on-chain audit trail for a significant governance transition. Off-chain systems (monitoring, governance dashboards, incident response) commonly rely on events to track role changes, and the lack of an event makes such changes harder to detect in real time.

The function is properly access-controlled (only the current deity may call it), but the transparency requirement remains unaddressed. Other parts of the contract do emit events (e.g., `NewTellorAddress` in the constructor), showing event usage is expected elsewhere. Not emitting a dedicated event for this role change is a logging/observability weakness.

Listing 1. Code snippet from contracts/TellorMaster.sol

```
54 function changeDeity(address _newDeity) external {
55     require(msg.sender == addresses[_DEITY]);
56     addresses[_DEITY] = _newDeity;
57 }
```

Go back to Findings Summary

L2: Missing event emission for owner change (changeOwner)

Impact	Low	Confidence	High
Target	contracts/TellorMaster.sol	Detection	Wake AI

Description

The contract allows the owner to be changed via `changeOwner` without emitting any event (e.g., a standard `OwnershipTransferred`). Changing ownership is a critical governance action, and the lack of a corresponding event degrades transparency and hampers off-chain monitoring and incident response.

Access control is correctly enforced by verifying `msg.sender == addresses[_OWNER]`, yet no on-chain log exists to reflect the change. This is a logging/observability weakness rather than a direct exploit.

Listing 2. Code snippet from contracts/TellorMaster.sol

```
63 function changeOwner(address _newOwner) external {
64     require(msg.sender == addresses[_OWNER]);
65     addresses[_OWNER] = _newOwner;
66 }
```

Go back to Findings Summary

”Static analysis or stay rekt”

https://wakehacker.ai