

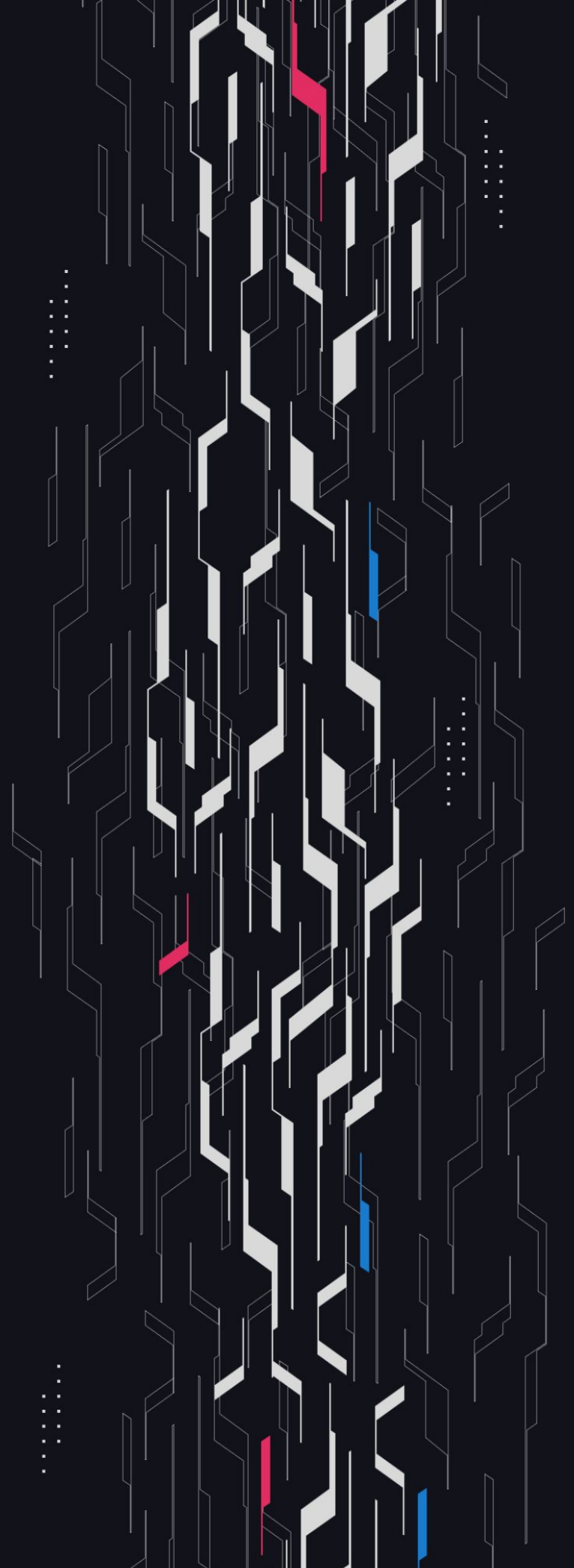
GA GUARDIAN

Trueo

**Decentralized
Prediction Market**

Security Assessment

March 8th, 2025



Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Nicholas Chew, Vladimir Zotov,

0xCiphky, Said, Ubermensch

Client Firm Trueo

Final Report Date March 8, 2025


Audit Summary

Trueo engaged Guardian to review the security of their decentralized prediction market platform, enabling users to forecast outcomes of real world events. From the 13th of January to the 27th of January, a team of 6 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 7 High/Critical issues were uncovered and promptly remediated by the Trueo team.

Security Recommendation Given the number of High and Critical issues detected as well as additional code changes made after the main review, Guardian recommends that an independent security review of the protocol at a finalized frozen commit is conducted before deployment.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

 Blockchain network: **Base**

 Verify the authenticity of this report on Guardian's GitHub: <https://github.com/guardianaudits>

 Code coverage & PoC test suite: <https://github.com/GuardianAudits/truth-markets-fuzzing>

Table of Contents

Project Information

Project Overview 4

Audit Scope & Methodology 5

Smart Contract Risk Assessment

Invariants Assessed 7

Findings & Resolutions 9

Addendum

Disclaimer 51

About Guardian Audits 52

Project Overview

Project Summary

Project Name	Trueo
Language	Solidity
Codebase	https://github.com/truth-market/truth-contracts
Commit(s)	Initial commit: 5c7d2eaafe1d120151be589ea0e42707df70cdd6 Final commit: b44c9834d6cf3ff93e949586dcff54190db94b23

Audit Summary

Delivery Date	March 8, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	1	0	0	0	0	1
● High	6	0	0	2	0	4
● Medium	13	0	0	6	2	5
● Low	19	0	0	19	0	0

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: <i>High</i>	Impact: <i>Medium</i>	Impact: <i>Low</i>
Likelihood: <i>High</i>	● Critical	● High	● Medium
Likelihood: <i>Medium</i>	● High	● Medium	● Low
Likelihood: <i>Low</i>	● Medium	● Low	● Low

Impact

- High** Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.
- Medium** A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.
- Low** Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

- High** The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.
- Medium** An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.
- Low** Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.
Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of Trueo, fuzz-testing with [Echidna](#) was performed on the protocol's main functionalities. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 50,000,000+ runs with a prepared Echidna fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
ERR-01	Only protocol errors allowed	✓	✓	✓	50M+
GLOB-00	Market status order violated	✓	✗	✓	50M+
GLOB-01	After a market reaches status Finalized, it's status should never change.	✓	✓	✓	50M+
GLOB-02	Should not see any ERC20InsufficientBalance reverts (specifically from the contract, actor balances should be handled with a sufficient mint and handler pre-conditions)	✓	✓	✓	50M+
GLOB-03	If MarketStatus == ResetByCouncil then oracleCouncil.getLastClosedDispute(address(this)).lastDisputorAddress != address(0)	✓	✓	✓	50M+
GLOB-04	TruthMarket should only hold a rewardToken balance of rewardAmount or 0	✓	✓	✓	50M+
GLOB-05	Payment token balance inside the market should always be enough to cover all user's burn under all market conditions.	✓	✓	✓	50M+
GLOB-06	EscalatedDisputerBondAmount should be equal to disputerBondAmount	✓	✓	✓	50M+
GLOB-07	DisputorTotalBond should be equal to disputorsCount * disputerBondAmount	✓	✓	✓	50M+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
GLOB-08	Resolution proposed before secondChallengePeriod elapsed after ResetByCouncil	✓	✗	✗	50M+
DIS-01	OracleCouncil.voteForDispute - disputeVotesCount should never change if the council member had a prior vote and is changing their vote.	✓	✓	✓	50M+
RES-01	After a proposeResolution function call the marketBond[_market].totalMarketBond should increase by the ITruthMarket(_marketAddress).resolverBondAmount()	✓	✗	✓	50M+
RES-02	After a proposeResolution function call the marketBond[_market].totalDepositedMarketBond should increase by the ITruthMarket(_marketAddress).resolverBondAmount()	✓	✗	✓	50M+
RES-03	OracleBonds tracking of each market's resolverBond balance should never exceed resolverBond amount. Any violation would imply multiple resolutions proposed or failure to refund bond during reset.	✓	✓	✓	50M+
RES-04	OracleBonds tracking of each market's escalatedDisputorBond balance should never exceed escalatorBondAmount amount. Any violation would imply multiple escalations possible.	✓	✓	✓	50M+
REED-01	Payment token balance inside the market should always be enough to cover all user's redeem when market is finalized and winning position is either YES or NO.	✓	✓	✓	50M+
WFCM-01	Payment token balance inside the market should always be enough to cover all user's withdrawFromCanceledMarket when market is finalized and winning position is CANCELED.	✓	✓	✓	50M+

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	New Resolver Loses Bond After Council Reset	Logical Error	● Critical	Resolved
H-01	Second Challenge Period Can Be Bypassed	Logical Error	● High	Acknowledged
H-02	Resolver Might Not Receive The Deserved Reward	Logical Error	● High	Resolved
H-03	Old Disputes Can Be Reused After ResetByCouncil	Logical Error	● High	Resolved
H-04	Unrestricted burn Could Lead To Unsettled Bonds	Logical Error	● High	Resolved
H-05	Locked Dispute Bonds After Escalation Reset	DOS	● High	Resolved
H-06	Token Holder Vote Cannot Be Disputed	Logical Error	● High	Acknowledged
M-01	NoToken Cap May Be Exceeded During Minting	Logical Error	● Medium	Resolved
M-02	Council Members Cannot Set Challenge Period	Access Control	● Medium	Resolved
M-03	Possible DOS In _getTotalOpenDisputes	DOS	● Medium	Acknowledged
M-04	Owner Cannot Trigger disputeMarket	Access Control	● Medium	Resolved
M-05	Inefficient Distribution In StakingRewards	Logical Error	● Medium	Acknowledged
M-06	Invalid Outcome Voting In voteForDispute	DOS	● Medium	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
M-07	OracleBonds Address Change Mishandles Funds	Logical Error	● Medium	Partially Resolved
M-08	Lacking Incentives For The Escalator	Logical Error	● Medium	Acknowledged
M-09	Disputer Cannot Re-Dispute After Reset	DOS	● Medium	Acknowledged
M-10	DoS In Dispute Process USDC Is Paused	DOS	● Medium	Acknowledged
M-11	Risk Of Re-org Attack	Reorg	● Medium	Acknowledged
M-12	USDC Blacklisted Users Can Freeze Market	Transfer	● Medium	Partially Resolved
M-13	Market DOS From Direct Reset Or Resolve	DOS	● Medium	Resolved
L-01	Markets Can Be Created And Cancelled For Profit	Logical Error	● Low	Acknowledged
L-02	Bond Amounts Not Validated	Validation	● Low	Acknowledged
L-03	Market Creation Lacks Input Validation	Validation	● Low	Acknowledged
L-04	Potential Griefing To Mint Transactions	Griefing	● Low	Acknowledged
L-05	Access Control Naming Convention	Best Practices	● Low	Acknowledged
L-06	Incorrect TrueToken maxSupply	Logical Error	● Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-07	Block In _isValidTransition Never Triggered	Logical Error	● Low	Acknowledged
L-08	Token Loss Via Truncation In Burn() Function	Truncation	● Low	Acknowledged
L-09	Clear Data After Removing Council Member	Logical Error	● Low	Acknowledged
L-10	Mismatched Event Parameter Order	Events	● Low	Acknowledged
L-11	Clear Data After Removing Pauser	Logical Error	● Low	Acknowledged
L-12	OracleCouncil Can't Pause/Unpause MarketManager	Logical Error	● Low	Acknowledged
L-13	setEndOfTrading Lacks Input Validation	Validation	● Low	Acknowledged
L-14	Missing setPaused Function In Market Manager	Logical Error	● Low	Acknowledged
L-15	Incorrect createdAt Timestamp In Escalations	Logical Error	● Low	Acknowledged
L-16	resolveEscalatedDispute Lacks Input Validation	Validation	● Low	Acknowledged
L-17	Disputes Allow Potential Abuse	Validation	● Low	Acknowledged
L-18	Reset By Council Cannot Be Escalated	Logical Error	● Low	Acknowledged
L-19	Max Council Members Can Be Violated	Logical Error	● Low	Acknowledged

C-01 | New Resolver Loses Bond After Council Reset

Category	Severity	Location	Status
Logical Error	● Critical	OracleBonds.sol: 120	Resolved

Description [PoC](#)

In the `sendResolverBondToMarket` function, the resolver bond is currently stored using:
`marketBond[_market].resolverBond = _amount;`

This approach overwrites any existing resolver bond, which is generally acceptable as only one resolution is expected.

However, an issue arises when a market status is `resetByCouncil`. Here's the problematic sequence:

1. Reset State: The council votes to reset the market, leaving the previous resolver's bond intact but not immediately returned.
2. New Proposal: A new resolver proposes a resolution, adding their bond via `sendResolverBondToMarket`.
3. Bond Overwrite: The `resolverBond` value is overwritten with the new bond, effectively discarding the previous bond value.
4. Loss of Funds: When the previous bond is sent to the disputor (to punish original resolver), the `resolverBond` is reset to zero, causing the new resolver to lose their bond.

Recommendation

To handle the situation where multiple bonds coexist during a reset, the resolver bonds should be stored incrementally and decremented appropriately when withdrawn or refunded.

Modify `sendResolverBondToMarket` to increment the value by the new bond amount:
`marketBond[_market].resolverBond = _amount.`

Similarly, in `sendBondFromMarketToSafeBox` and `issueBondsBackToResolver`, decrement the resolver bond by the transferred amount instead of setting it to zero.

Resolution

Trueo Team: Resolved.

H-01 | Second Challenge Period Can Be Bypassed

Category	Severity	Location	Status
Logical Error	● High	TruthMarket.sol: 199-209	Acknowledged

Description [PoC](#)

When the market is in the `MarketStatus.ResetByCouncil` state, it should wait for the `secondChallengePeriod` before opening for resolution to allow `openEscalatedDispute`. However, `proposeResolution` can be called immediately without waiting for the `secondChallengePeriod`.

The bypass of `secondChallengePeriod` allows a dishonest disputor to move the market status to `ResolutionProposed` therefore earning the resolver's bond before the resolver has a chance to escalate the issue.

Recommendation

Consider moving `_updateStatus` execution before setting `councilDecisionAt = 0`.

Resolution

Trueo Team: Acknowledged.

H-02 | Resolver Might Not Receive The Deserved Reward

Category	Severity	Location	Status
Logical Error	● High	TruthMarket.sol: 523-525, TruthMarket.sol: 549-551	Resolved

Description

When a market is finalized and ends with `_CANCELED` as the `winningPosition`, users can trigger `withdrawFromCanceledMarket` to withdraw their payment tokens.

Within `withdrawFromCanceledMarket`, if `bondSettled` is not yet settled, it will trigger `_settleBonds` to handle reward payments and bond refunds or slashes.

However, if the market previously entered a dispute or escalation state, it will transfer the reward to the `safeBoxAddress` when the `winningPosition` is `_CANCELED`, regardless of whether the `originalOutcomeFromResolver` is equal to `_CANCELED`.

This will result in the resolver not receiving the deserved reward.

Recommendation

Only transfer reward to `safeBoxAddress` when `winningPosition` is `_CANCELED` and `winningPosition` is not equal to `originalOutcomeFromResolver`

Resolution

Trueo Team: Resolved.

H-03 | Old Disputes Can Be Reused After ResetByCouncil

Category	Severity	Location	Status
Logical Error	● High	TruthMarket.sol: 199-209	Resolved

Description [PoC](#)

When a market is partially disputed, multiple disputes can be opened in parallel. If one of those disputes concludes first (e.g., triggering a `ResetByCouncil`), the market transitions back to `OpenForResolution`, and any unclosed disputes remain in storage, still carrying prior votes.

Because those old disputes are never marked as permanently invalid, they can be reused when the market enters a new resolution flow—allowing a malicious council member to “revive” the partially-voted dispute and finalize it with minimal new votes.

In the provided example:

1. A market transitions to `ResolutionProposed` with a “YES” outcome.
2. Two different disputers each open a separate dispute (Dispute #1 and Dispute #2).
3. Dispute #1 closes first (for example, by a majority that sets `ResetByCouncil`), while Dispute #2 remains open but unresolvable at that moment due to the market status update.
4. The market then resets, returning to `OpenForResolution` (or even `ResolutionProposed` again).
5. A malicious council member can propose a new outcome, open a fresh dispute, but then re-invoke the old, unresolved Dispute #2. Because that older dispute already has partial votes recorded, the malicious user can cast a single new vote (or minimal votes) to reach a majority, effectively closing Dispute #2 and imposing its outcome—even though other council members haven’t re-voted under the new context.

Recommendation

Invalidate or close all existing disputes when the market transitions from `SetByCouncil` or `ResetByCouncil` back to `OpenForResolution`.

Resolution

Trueo Team: Resolved.

Guardian Team: The new check if `(_disputeIndex <= marketLastClosedDispute[_market]) revert DisputeInvalid()`; is ineffective because an old dispute may have a higher index than the last closed dispute.

H-04 | Unrestricted burn Could Lead To Unsettled Bonds

Category	Severity	Location	Status
Logical Error	● High	TruthMarket.sol: 324-330	Resolved

Description

burn can be called by anyone at any market state, allowing users to burn their yes and no tokens to retrieve the deposit tokens.

This could cause issues in a scenario where the market is finalized, especially when the winning position is _CANCELED. Instead of calling withdrawFromCanceledMarket, users might choose to call burn to reclaim their deposit tokens.

This would prevent _settleBonds from being triggered, causing all bonds and rewards to remain stuck and unresolved.

Recommendation

Consider restricting burn so it can only be called when the market is not finalized. Additionally, make settle bonds publicly callable in cases where no token holders decide to redeem or withdraw their tokens.

Resolution

Trueo Team: Resolved.

H-05 | Locked Dispute Bonds After Escalation Reset

Category	Severity	Location	Status
DOS	● High	OracleCouncilV2.sol: 421	Resolved

Description [PoC](#)

When a dispute is resolved by the council and its escalation leads to a Reset result, any user with an “unclosed dispute” bond remains unable to claim it because `reopenMarketForDisputes` sets `marketClosedForDisputes` to false.

Consequently, once the market transitions to a new proposal, if it finalizes with no disputes, the dispute is never recognized as “closed” or “canceled,” preventing `claimUnclosedDisputeBonds` from succeeding. The disputer’s bond remains locked, and they are unable to recover their funds.

Recommendation

Update `canDisputorClaimbackBondFromUnclosedDispute` to allow bond retrieval if the market has been reset, rather than strictly requiring it to be “closed for disputes” or “finalized.” This ensures no user is stuck with an unclaimable dispute bond after the escalation reset.

Resolution

Trueo Team: Resolved.

H-06 | Token Holder Vote Cannot Be Disputed

Category	Severity	Location	Status
Logical Error	● High	Global	Acknowledged

Description

The current implementation dictates that once an escalation is decided by a token holders' vote, the market either resolves to Finalized or reverts to `OpenForResolution`, with no mechanism to dispute this decision.

However, according to the documentation, a third challenge window should exist, enabling any holder of 250,000 TRUE tokens to dispute the escalation result and escalate the matter to the Attesters.

The absence of this challenge window introduces a vulnerability where large TRUE token holders could manipulate the vote for financial gain, leaving other stakeholders without any recourse to contest the outcome.

Recommendation

Introduce a third challenge window as outlined in the documentation, ensuring that escalation decisions can be disputed and reviewed by the Attesters.

Resolution

Trueo Team: Acknowledged.

M-01 | NoToken Cap May Be Exceeded During Minting

Category	Severity	Location	Status
Logical Error	● Medium	TruthMarket.sol: 312	Resolved

Description

The mint function enforces the `yesNoTokenCap`, but it only checks the supply of Yes tokens. This creates a potential issue where the supply of No tokens can exceed the cap if there is an imbalance between the total supply of Yes and No tokens.

This imbalance can occur because Yes tokens can be burned (since `YesNoToken` is an `ERC20Burnable` contract).

For example:

- `yesNoTokenCap` = 100
- Initial supply: Yes = 90, No = 90
- A user burns 10 Yes tokens, reducing the Yes supply to 80.
- The same user mints 20 Yes tokens, bringing the Yes supply back to 100.
- The final supply becomes: Yes = 100, No = 110, exceeding the cap.

Recommendation

Check that the No token supply does not exceed the `yesNoTokenCap` during minting.

Resolution

Trueo Team: Resolved.

M-02 | Council Members Cannot Set Challenge Period

Category	Severity	Location	Status
Access Control	● Medium	TruthMarketManagerV2.sol: 355-361	Resolved

Description

Inside TruthMarketManagerV2, several functions are available to manage the market's configuration that can be called by the oracle council, such as setYesNoTokenCap, setEndOfTrading, setFirstChallengePeriod, and setSecondChallengePeriod.

While setYesNoTokenCap and setEndOfTrading are currently available in the OracleCouncilV2 contract, setFirstChallengePeriod and setSecondChallengePeriod are not implemented, preventing council members from configuring those market settings.

Recommendation

Implement setFirstChallengePeriod and setSecondChallengePeriod inside OracleCouncilV2.

Resolution

Trueo Team: Resolved.

M-03 | Possible DOS In _getTotalOpenDisputes

Category	Severity	Location	Status
DOS	● Medium	OracleCouncilV2.sol: 536	Acknowledged

Description

The `_getTotalOpenDisputes` function is used by the `addOracleCouncilMember` and `removeOracleCouncilMember` functions in the `OracleCouncil` contract.

This function calculates the total open disputes by looping through all active markets and checking if each market is in the `DisputeRaised` status and has any open disputes. The issue is that `_activeMarkets` does not have a cap, meaning it could grow indefinitely.

As a result, the gas costs for calling `_getTotalOpenDisputes` can become excessively high, potentially exceeding the block gas limit and causing the transaction to fail.

This could effectively lock the `addOracleCouncilMember` and `removeOracleCouncilMember` functions, preventing them from being executed.

Recommendation

Introduce a cap on the size of `_activeMarkets` or consider optimizing the function to avoid iterating through markets that are in a finalized state, as they cannot return to the disputed state.

Resolution

Trueo Team: Acknowledged.

M-04 | Owner Cannot Trigger disputeMarket

Category	Severity	Location	Status
Access Control	● Medium	TruthMarketManagerV2.sol: 303-318	Resolved

Description

Many operations inside TruthMarketManagerV2 that change the market state can be manually triggered by the owner. This is needed in cases where markets are paused, as only the owner can initiate market state transitions, including disputeMarket.

However, disputeMarket uses the onlyOracleCouncil modifier, which restricts the function to being called only by the oracle council. This is despite the function logic including a check allowing the owner to trigger disputeMarket when markets are paused.

As a result, disputeMarket cannot be manually called by the owner when markets are paused.

Recommendation

Consider to use onlyOracleCouncilAndOwner instead.

Resolution

Trueo Team: Resolved.

M-05 | Inefficient Distribution In StakingRewards

Category	Severity	Location	Status
Logical Error	● Medium	StakingRewards.sol	Acknowledged

Description [PoC](#)

If `stake` is not called in the same block of `notifyRewardAmount`, depending on delay, a portion of rewards will remain unused inside the contract.

For example, at time `X` rewards are transferred into the contract. Then some time `Y` has passed before the first user stakes. However, the reward period will end at `X + rewardsDuration` not `X + Y + rewardsDuration`.

Therefore, the rewards for `Y * rewardRate` will remain un-distributed till the next cycle. If a new reward cycle is never started (e.g. final cycle), then any undistributed amount will remain inside the contract.

Recommendation

Consider defining `periodFinish` in the first `stake` that is done after `notifyRewardAmount`, when total deposits are zero.

Resolution

Trueo Team: Acknowledged.

M-06 | Invalid Outcome Voting In voteForDispute

Category	Severity	Location	Status
DOS	● Medium	OracleCouncilV2.sol: 187	Resolved

Description

Council members can cast votes referencing an invalid `_winningPosition`, one not recognized by the market. If enough votes align on this invalid outcome, the dispute remains unresolvable. Consequently, it remains open indefinitely, blocking certain system actions (e.g., adding or removing council members) which require all disputes to be closed.

Recommendation

Enforce strict `_winningPosition` validation in `voteForDispute`, reverting if `_winningPosition` falls outside 1 to `ITruthMarket(_market).positionCount()`.

Resolution

Trueo Team: Resolved.

M-07 | OracleBonds Address Change Mishandles Funds

Category	Severity	Location	Status
Logical Error	● Medium	TruthMarketManagerV2.sol: 496	Partially Resolved

Description

The `setOracleBonds` function in the `TruthMarketManager` contract allows the owner to change the `oracleBonds` address. The problem is that the `oracleBonds` address is hardcoded into markets when they are created.

If the address is updated in the `TruthMarketManager` contract, active markets not yet finalized will send and account for resolve, dispute, and escalate bonds to the new address.

However, when issuing them back, the old address in the market contract will be used, which will not have the funds or have them accounted for, leading to users losing those funds.

Furthermore, any unclaimed dispute bonds will also be unclaimable because the `OracleCouncil` contract will fetch the new address from the `TruthMarketManager` contract, while these dispute bonds are stored in and accounted for in the old contract.

Recommendation

If an `oracleBonds` update is needed, ensure it is performed only when all active markets are finalized and manually handle users' unclaimed disputed bonds, as the normal functionality will not work for pending disputes before the address change.

Resolution

Trueo Team: Resolved.

Guardian Team: The `setOracleBonds` function now verifies that all active markets have settled their bonds before an address update is allowed. However, even with this precaution, any unclaimed dispute bonds become unclaimable.

M-08 | Lacking Incentives For The Escalator

Category	Severity	Location	Status
Logical Error	● Medium	TruthMarket.sol: 523-529, TruthMarket.sol: 549-555	Acknowledged

Description

Currently, if the escalator is correct, the reward is sent to either the resolver or the disputer. This assumes that the escalator is also the resolver or disputer.

However, considering that the escalator requires a larger bond, it's possible that the resolver or disputer does not have the required bond, and another party could step in and become the escalator. This third party should be able to receive the reward independently for a correct escalation.

Recommendation

Consider splitting the reward between the disputer/resolver and escalator when escalator is correct.

Resolution

Trueo Team: Acknowledged.

M-09 | Disputer Cannot Re-Dispute After Reset

Category	Severity	Location	Status
DOS	● Medium	OracleCouncilV2.sol: 419-428	Acknowledged

Description

When a market is reset following multiple active disputes, a user whose dispute is still considered “open” cannot reclaim their bond because the system only allows bond retrieval after the market is finalized.

As a result, that user remains in a state with an unclaimed (open) dispute bond, which prevents them from initiating a new dispute on a subsequent outcome proposal. This essentially locks them out of further participation in the dispute process for that market.

Recommendation

Enable retrieval or reusability of the bond upon a market reset for disputes that remain open. Specifically, consider allowing users to claim or “migrate” their open dispute bond once the market transitions to a reset state—rather than strictly requiring the market to be finalized.

Resolution

Trueo Team: Acknowledged.

M-10 | DoS In Dispute Process USDC Is Paused

Category	Severity	Location	Status
DOS	● Medium	OracleBonds.sol: 301-311	Acknowledged

Description

The market’s dispute and escalation mechanisms rely on the ability to deposit bonds in USDC. If USDC—being a pausable token—is paused, no new bonds can be deposited, effectively blocking any new disputes or escalations.

An attacker can exploit this by proposing a resolution in their favor just before USDC becomes paused; if the pause period extends past the challenge window, the outcome remains uncontested, leading to unjust financial gains for the attacker.

Recommendation

Extend dispute periods when USDC (or any payment token) is paused until transfers become possible again.

Resolution

Trueo Team: Acknowledged.

M-11 | Risk Of Re-org Attack

Category	Severity	Location	Status
Reorg	● Medium	Global	Acknowledged

Description

In the case of a block re-org event, disputors may unknowingly have submitted disputes for what they believe to be the correct result, leading to slashing and the loss of their disputer bonds.

For example, consider the following scenario:

- (1) Alice proposes an incorrect resolution.
- (2) Bob disputes Alice’s resolution.
- (3) A block re-org occurs and Alice’s proposal is replaced with a correct resolution.
- (4) Bob’s dispute is invalid and will be punished.

Recommendation

Allow disputors to pass the exact position they’d like to dispute to function `openDispute`.

Resolution

Trueo Team: Acknowledged.

M-12 | USDC Blacklisted Users Can Freeze Market

Category	Severity	Location	Status
Transfer	● Medium	TruthMarket.sol	Partially Resolved

Description

If the market is disputed or escalated, it could transfer bonds back to the disputer and escalator if they are not punished. However, since the `paymentToken` is USDC, which has a blacklist feature, it is possible for the disputer or escalator is blacklisted when the market attempts to return the bonds.

This would cause the market state transitions to fail. The same issue could occur when attempting to send rewards to the resolver or disputer if the reward token also has a blacklist feature.

Recommendation

Consider using the Pull-over-Push pattern when handling bonds and rewards.

Resolution

Trueo Team: Resolved.

Guardian Team: The transfer could still be blocked when sending rewards to a blacklisted address.

M-13 | Market DOS From Direct Reset Or Resolve

Category	Severity	Location	Status
DOS	● Medium	TruthMarketManagerV2.sol: 257	Resolved

Description [PoC](#)

A market can be resolved or reset by the owner directly from the `TruthMarketManager` contract. One instance where this will be used is, if a market is paused, only the owner can call the reset or resolve market functions instead of following the general flow through either the `OracleCouncil` contract or the `Escalation` contract.

The issue is that when called directly, the dispute or escalation processes are not fully completed as they would be in the general flow. In the dispute case, for instance, one issue would be the `marketLastClosedDispute` not being updated, meaning the `isResolverPunished` and `isDisputorPunished` flags remain unset, defaulting to `false`.

Consequently, when the market either resets or finalizes, it will attempt to send funds to the disputer's address fetching the address from the unset `marketLastClosedDispute`. Since this address is not set, it would return the zero address, leading to a revert and locking up the market.

Furthermore, the `marketClosedForDisputes` is also not set to `true` as done in the general flow, meaning unclaimed disputes for the market will remain locked.

Similarly, in the escalation case, if called directly for either reset or resolve, the lack of updates to `marketToEscalatedDispute` will leave multiple variables such as punishment outcomes empty.

This will also cause incorrect outcomes, as the punishment outcomes will all default to `false` and possibly other issues.

Recommendation

In the current code, a direct call to reset or resolve the market will cause multiple issues, potentially locking up the market and user funds. Either modify the logic to ensure that if called directly, the dispute or escalation processes are still fully completed, or avoid calling them directly entirely.

Resolution

Trueo Team: Resolved.

L-01 | Markets Can Be Created And Cancelled For Profit

Category	Severity	Location	Status
Logical Error	● Low	Global	Acknowledged

Description

In the current implementation, market creation is restricted to trusted council members. However, once market creation becomes permissionless as per the documentation, it introduces the risk of abuse.

Malicious actors could create nonsensical or deliberately unresolvable markets and then submit a CANCELED resolution to claim resolver fees

Recommendation

To mitigate this risk consider implementing a market creation fee when market creation becomes permissionless.

Resolution

Trueo Team: Acknowledged.

L-02 | Bond Amounts Not Validated

Category	Severity	Location	Status
Validation	● Low	TruthMarketManagerV2.sol: 465	Acknowledged

Description

Within function `setAmounts` in the `TruthMarketManager`, is it not validated that `escalatorBondAmount` is the largest bond amount relative to the disputer and resolver bond amounts.

According to documentation, "Every escalation must come at an increased cost to that of the previous, so that the protocol may potentially slash the bonds in compensation for the expended resources."

Recommendation

Validate the bonds amounts to be according to documentation.

Resolution

Trueo Team: Acknowledged.

L-03 | Market Creation Lacks Input Validation

Category	Severity	Location	Status
Validation	● Low	TruthMarketManagerV2.sol: 176	Acknowledged

Description

Currently, the `createMarket` function lacks validation for several input fields, which could lead to the creation of invalid or exploitable markets.

Specifically:

- `rewardToken` should be a non-zero address
- `rewardAmount` should not be zero and have a minimum amount
- `yesNoTokenCap` should be of a reasonable amount

Recommendation

Implement validations for the input fields described above.

Resolution

Trueo Team: Acknowledged.

L-04 | Potential Griefing To Mint Transactions

Category	Severity	Location	Status
Griefing	● Low	TruthMarket.sol: 303-330	Acknowledged

Description

The mint function is exposed to a griefing attack where an attacker can frontrun a legitimate user's transaction by minting the exact remaining tokens under the yesNoTokenCap. This causes the legitimate user's transaction to revert due to the TokenCapExceeded() check.

The attacker can then backrun with a burn transaction, reclaiming the payment tokens spent during the frontrun. Although this attack requires the attacker to have sufficient capital to mint the remaining tokens, it effectively disrupts legitimate users.

Recommendation

Introduce a mechanism to limit how quickly addresses can mint, or limit the maximum mint amount per transaction.

Resolution

Trueo Team: Acknowledged.

L-05 | Access Control Naming Convention

Category	Severity	Location	Status
Best Practices	● Low	OraclePausable.sol	Acknowledged

Description

Within OraclePausable, the modifier pauserOnly breaks the typically access control naming convention of onlyRole

Recommendation

Consider updating the modifier to onlyPauser.

Resolution

Trueo Team: Acknowledged.

L-06 | Incorrect TrueToken maxSupply

Category	Severity	Location	Status
Logical Error	● Low	TrueToken.sol: 11	Acknowledged

Description

The Truemarket documentation states that the total supply of TRUE is 100 million. However, the current maxSupply set in the TrueToken contract is 1 billion, which contradicts the documentation.

Recommendation

Update the maxSupply value in the TrueToken contract to align with the documented total supply of 100 million.

Resolution

Trueo Team: Acknowledged.

L-07 | Block In _isValidTransition Never Triggered

Category	Severity	Location	Status
Logical Error	● Low	TruthMarket.sol: 600	Acknowledged

Description

In the `_isValidTransition` function, the `else` block for handling transitions when the from status is `ResetByCouncil` is never triggered.

This is due to the behavior of `getCurrentStatus`, which automatically transitions the `ResetByCouncil` status to `OpenForResolution` if the current timestamp is past the second challenge period.

Furthermore, the logic in the `else` block is incorrect because, there is no direct transition from `ResetByCouncil` to `OpenForResolution`.

Recommendation

Consider removing the `else` block.

Resolution

Trueo Team: Acknowledged.

L-08 | Token Loss Via Truncation In Burn() Function

Category	Severity	Location	Status
Truncation	● Low	TruthMarket.sol: 324-330	Acknowledged

Description

When `amount` is multiplied by $10^{\text{paymentTokenDecimals}}$ and then divided by $10^{\text{tokenDecimals}}$, any `amount` smaller than the ratio $10^{(\text{tokenDecimals} - \text{paymentTokenDecimals})}$ is rounded down to 0.

As a result, users may burn their YES/NO tokens but receive zero payment tokens, causing them to lose tokens unintentionally.

Recommendation

Require `paymentTokenAmount > 0` before executing the burn, it's also recommended to only burn a multiple of $10^{(\text{tokenDecimals} - \text{paymentTokenDecimals})}$ from the users token avoiding loss from their end due to truncation.

Resolution

Trueo Team: Acknowledged.

L-09 | Clear Data After Removing Council Member

Category	Severity	Location	Status
Logical Error	● Low	OracleCouncilV2.sol: 136-146	Acknowledged

Description

When `removeOracleCouncilMember` is called, it moves `councilMemberAddress` at `councilMemberCount` to `councilMemberIndex[_councilMember]`, but the `councilMemberAddress` at `councilMemberCount` is not cleared.

Recommendation

When removing a council member, consider clearing `councilMemberAddress` at `councilMemberCount`.

Resolution

Trueo Team: Acknowledged.

L-10 | Mismatched Event Parameter Order

Category	Severity	Location	Status
Events	● Low	TruthMarketManagerV2.sol: 396-404	Acknowledged

Description

In the `setAddresses` function, the `AddressesUpdated` event is emitted with a specific parameter order. However, the actual arguments passed do not match this order.

This causes misalignment between the event’s named parameters and the actual addresses being updated, leading to potential confusion or incorrect off-chain tracking.

Recommendation

Reorder the arguments in the `emit AddressesUpdated(...)` call to match the event’s parameter list or update the event definition to align with the actual argument order.

Resolution

Trueo Team: Acknowledged.

L-11 | Clear Data After Removing Pauser

Category	Severity	Location	Status
Logical Error	● Low	TruthMarketManagerV2.sol: 526-538	Acknowledged

Description

When `removePauserAddress` is called, it moves `pauserAddress` at `pauserCount` to `pauserIndex[_pauserAddress]`, but the `pauserAddress` at `pauserCount` is not cleared.

Recommendation

When a pauser is removed, consider clearing `pauserAddress` at `pauserCount`

Resolution

Trueo Team: Acknowledged.

L-12 | OracleCouncil Can't Pause/Unpause MarketManager

Category	Severity	Location	Status
Logical Error	● Low	TruthMarketManagerV2.sol: 540	Acknowledged

Description

The TruthMarketManager contract uses the onlyOracleCouncilAndOwner modifier on the pause and unpause functions, allowing the owner and OracleCouncil address to call them.

However, the OracleCouncil contract does not implement the functionality to invoke these functions, preventing it from pausing or unpausing the TruthMarketManager contract as intended.

Recommendation

Implement the necessary functionality in the OracleCouncil contract to allow it to call the pause and unpause functions in the TruthMarketManager contract.

Resolution

Trueo Team: Acknowledged.

L-13 | setEndOfTrading Lacks Input Validation

Category	Severity	Location	Status
Validation	● Low	TruthMarketManagerV2.sol: 351-353	Acknowledged

Description

When a market is newly created, it ensures that `_endOfTrading` is greater than `minimumTradingDuration`. However, when `setEndOfTrading` is called, there is no validation, making it possible to configure the market with an `endOfTrading` that is less than `minimumTradingDuration`.

Recommendation

Add the same validation inside `setEndOfTrading`.

Resolution

Trueo Team: Acknowledged.

L-14 | Missing setPaused Function In Market Manager

Category	Severity	Location	Status
Logical Error	● Low	TruthMarketManagerV2.sol	Acknowledged

Description

In OraclePausable, setPaused can be called by owner() which is the TruthMarketManagerV2 contract. However, this function is not implemented in TruthMarketManagerV2.

Recommendation

Consider implementing setPaused in TruthMarketManagerV2.

Resolution

Trueo Team: Acknowledged.

L-15 | Incorrect createdAt Timestamp In Escalations

Category	Severity	Location	Status
Logical Error	● Low	Escalation.sol: 136	Acknowledged

Description

When a new dispute is opened through the `openEscalatedDispute` function, the dispute's `createdAt` is set to `block.timestamp`.

However, this value is overwritten when the `setEscalationProposalId` function is called, which sets `createdAt` to `block.timestamp` again. As a result, the `createdAt` variable will not accurately reflect the original time the dispute was created.

Recommendation

Ensure that the `createdAt` variable is only set once when the dispute is initially opened through the `openEscalatedDispute`.

Resolution

Trueo Team: Acknowledged.

L-16 | resolveEscalatedDispute Lacks Input Validation

Category	Severity	Location	Status
Validation	● Low	Escalation.sol	Acknowledged

Description

In the function `resolveEscalatedDispute`, the input arguments should be validated to avoid any illogical state.

Notably:

- `_isResultReset` and `_isResultAccept` should never be equal
- Punish arguments (x3) should never all be `true`.

Additionally, `isEscalatedDisputorPunished` should be set to `false` when `_isResultAccept` is true - as the escalator should never be punished if the escalation was accepted.

Similarly, `isCouncilDisputorPunished` should be set to `false` when `_isResultAccept` is false. Punish arguments should be validated to ensure they do not conflict with this.

Recommendation

Add the recommended validations to `resolveEscalatedDispute`.

Resolution

Trueo Team: Acknowledged.

L-17 | Disputes Allow Potential Abuse

Category	Severity	Location	Status
Validation	● Low	OracleCouncilV2.sol: 148	Acknowledged

Description

Users can open a dispute for a market once a resolution is proposed by calling the `openDispute` function with the market address and `_disputeString` as parameters. The `_disputeString` allows users to explain the reason for the dispute and present evidence.

However, with no restrictions or validations on the `_disputeString`, malicious users could exploit this feature to post malicious or misleading links. If these links are displayed on the front end, they could lead unsuspecting users to phishing sites or other malicious content.

Additionally, since only one dispute can be processed and rewarded or punished, the cost of such an attack is limited to the disputer bond amount. A malicious actor could open multiple disputes with different dispute strings to increase the chances of unsuspecting users clicking on the harmful links.

Recommendation

Ensure that disputes displayed on the front end are validated to prevent malicious content from being presented to users.

Resolution

Trueo Team: Acknowledged.

L-18 | Reset By Council Cannot Be Escalated

Category	Severity	Location	Status
Logical Error	● Low	Trueo.sol: 228	Acknowledged

Description

When the council decides to reset a market with `_returnToOpenForResolution == true`, market status is immediately set to `OpenForResolution`.

As a result, the decision cannot be disputed by escalation which prevents a confident resolver from disputing the decision, so as not to lose the potential reward.

Recommendation

Consider if this is expected. Otherwise, allow for dispute by escalation, even if the council's decision is to reset and open market for resolution.

Resolution

Trueo Team: Acknowledged.

L-19 | Max Council Members Can Be Violated

Category	Severity	Location	Status
Logical Error	● Low	OracleCouncilV2.sol: 125	Acknowledged

Description

The `addOracleCouncilMember` function incorrectly implements the check for the maximum number of council members, allowing the limit to be exceeded by one.

The condition:

```
if (councilMemberCount > marketManager.maxOracleCouncilMembers()) revert  
MaxOracleCouncilMembersExceeded();
```

 does not prevent adding an 11th member when the maximum allowed is 10.

This happens because the check is only triggered after the count exceeds the limit, rather than when it reaches the limit.

Recommendation

Update the condition to ensure the council member count does not exceed the maximum:

```
if (councilMemberCount = marketManager.maxOracleCouncilMembers()) revert  
MaxOracleCouncilMembersExceeded();
```

Resolution

Trueo Team: Acknowledged.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit <https://guardianaudits.com>

To view our audit portfolio, visit <https://github.com/guardianaudits>

To book an audit, message <https://t.me/guardianaudits>