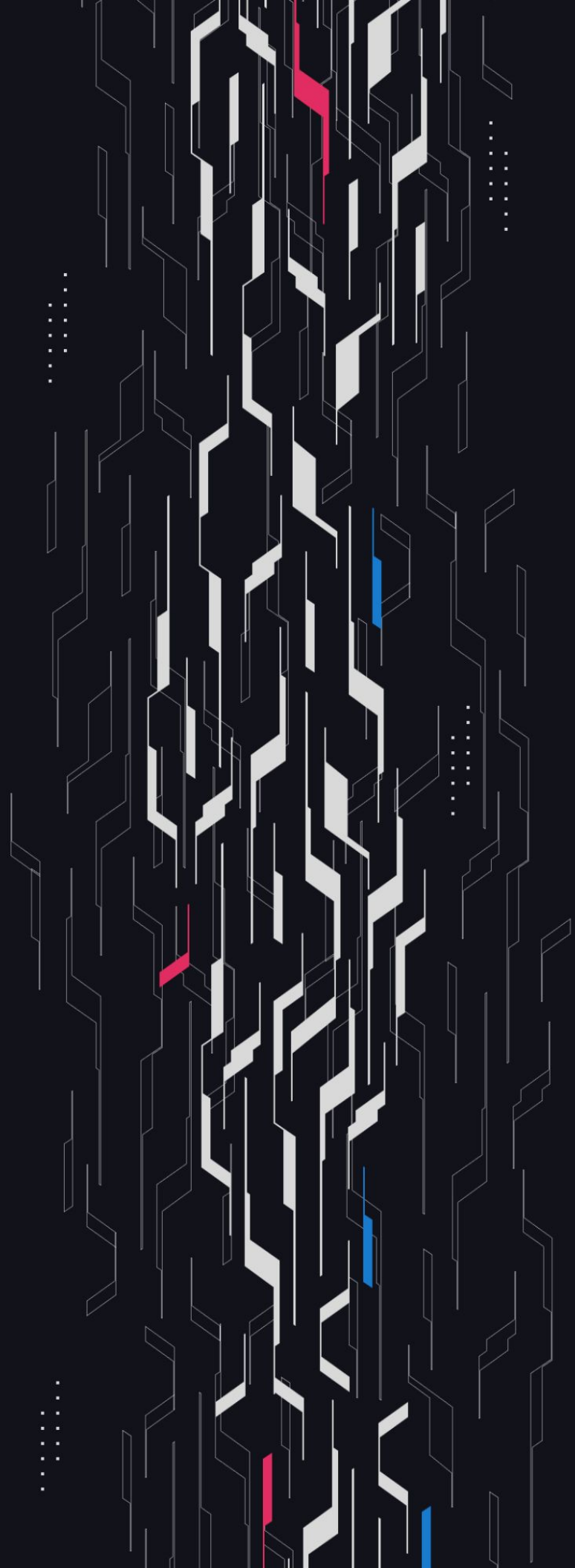


GA GUARDIAN

Yellow

Yellow

March 8th, 2026



Summary

Audit Firm Guardian

Prepared By Osman Ozdemir, Ali Shebab

Client Firm Yellow Network

Final Report Date March 8, 2026

Audit Summary

Yellow engaged Guardian to review the security of their Yellow contract. From the 2nd of March 2026 to the 5th of March 2026, a team of 2 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Confidence Ranking

Given the lack of High and Critical issues detected during the main review, Guardian assigns a Confidence Ranking of 5 to the protocol. Guardian advises the protocol to consider periodic review with future changes. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

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

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. Recommendation: Code is highly secure at time of audit. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	Code is clean, well-structured, and adheres to best practices. Only 1 Significant issue was uncovered per week. Design patterns are sound, and test coverage is strong. Recommendation: Suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity issues. Varied Low severity findings.
3: Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases. Recommendation: Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1-2 High/Critical findings per engagement week.
2: Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities. Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. Recommendation: Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week. Or additional High/Critical findings uncovered in remediation review which have not been resolved and confirmed by Guardian.
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥ 5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. Recommendation: Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥ 5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Information

Project Overview 5

Audit Scope & Methodology 6

Addendum

Disclaimer 19

About Guardian 20

Project Overview

Project Summary

Project Name	Yellow
Language	Solidity
Codebase	https://github.com/layer-3/yellow
Commit(s)	Main Review commit: a805c88cf5ce3c7eb46cdd5446dbed6d04c50f03

Audit Summary

Delivery Date	March 8, 2026
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	0	0	0	0	0	0
● Medium	1	0	0	0	0	1
● Low	4	0	0	0	0	4
● Info	4	0	0	0	0	4

Audit Scope & Methodology

```
contract,source,total,comment
yellow/src/Governor.sol,87,122,15
yellow/src/Locker.sol,75,125,21
yellow/src/Token.sol,11,23,8
yellow/src/Treasury.sol,30,51,13
source count: {
  total: 321,
  source: 203,
  comment: 57,
  single: 20,
  block: 37,
  mixed: 0,
  empty: 61,
  todo: 0,
  blockEmpty: 0,
  commentToSourceRatio: 0.28078817733990147
}
```

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.

Findings & Resolutions

ID	Title	Category	Severity	Status
M-01	Quorum Floor Change Affects Previous Proposals	Logical Error	● Medium	Resolved
L-01	Quorum Floor Can Be Set Above Voting Supply	Validation	● Low	Resolved
L-02	Undelegated Locks Raise Quorum With No Votes	Logical Error	● Low	Resolved
L-03	No Late-Quorum Protection On Governor	Best Practices	● Low	Resolved
L-04	Ownership Can Be Renounced And Cause Permanent DoS	Configuration	● Low	Resolved
I-01	Rebasing Tokens Cannot Be Used In Locker	Warning	● Info	Resolved
I-02	Mismatch Between Tests And Code Comment	Informational	● Info	Resolved
I-03	Proposals Uncancellable Once Past Pending State	Best Practices	● Info	Resolved
I-04	Misleading Error Name In Constructor Validation	Best Practices	● Info	Resolved

M-01 | Quorum Floor Change Affects Previous Proposals

Category	Severity	Location	Status
Logical Error	● Medium	Governor.sol	Resolved

Description

The quorum value is determined as the maximum of the fractional quorum and quorumFloor. While the fractional quorum, inherited from GovernorVotesQuorumFraction, is calculated using the snapshot at the time of proposal creation, quorumFloor uses the current value and is not snapshotted at proposal creation.

The queue and execute flows perform quorum checks based on updated value. As a result, any update to quorumFloor retroactively affects all proposals that have not yet been executed, including proposals whose voting period has already ended. A proposal may satisfy quorum and be marked as passed under the floor value in effect during voting, but a subsequent increase to quorumFloor can cause quorum validation to fail during the queuing or execution phase, preventing successful proposal execution.

Recommendation

Snapshot the quorum floor per proposal (store floor at proposal creation/snapshot) and use that stored value when computing quorum for that proposal’s snapshot block. Alternatively, make the floor change apply only to proposals created after a certain timepoint.

Resolution

Yellow Team: Resolved.

L-01 | Quorum Floor Can Be Set Above Voting Supply

Category	Severity	Location	Status
Validation	● Low	Governor.sol	Resolved

Description

Governance can set `_quorumFloor` to an absolute value larger than the maximum possible votes at a snapshot (e.g., larger than total voting units / total locked supply), making quorum impossible to reach and effectively freezing proposal success/execution. Because `setQuorumFloor` itself is `onlyGovernance`, a malicious or compromised governance moment can permanently brick future governance.

Recommendation

Add an upper bound when setting the floor (e.g., `<= token.totalSupply()` or `<= expected max voting units`).

Resolution

Yellow Team: Resolved.

L-02 | Undelegated Locks Raise Quorum With No Votes

Category	Severity	Location	Status
Logical Error	● Low	Locker.sol	Resolved

Description

In src/Locker.sol, lock(uint256 amount) calls _transferVotingUnits(address(0), msg.sender, received) regardless of delegation status. In OZ Votes, this unconditionally adds to _totalCheckpoints (the total supply used in quorum calculation).

However, when the user has not called delegate(), their delegatee defaults to address(0), so _moveDelegateVotes(address(0), address(0), amount) hits the from != to guard and is a no-op; no voting power is assigned to any address.

Recommendation

Consider auto-self-delegating on first lock when delegates(msg.sender) == address(0) (e.g., call _delegate(msg.sender, msg.sender) in lock()).

Resolution

Yellow Team: Resolved.

L-03 | No Late-Quorum Protection On Governor

Category	Severity	Location	Status
Best Practices	● Low	Governor.sol	Resolved

Description

YellowGovernor (src/Governor.sol) does not inherit OZ's GovernorPreventLateQuorum extension. Without it, a whale who holds sufficient voting power can wait until the final block of the voting period and cast a decisive vote that triggers quorum and determines the outcome in a single transaction. Other tokenholders have no opportunity to observe the vote, react, or counter-vote because the proposal deadline has already passed.

Recommendation

Consider inheriting GovernorPreventLateQuorum and setting a reasonable voteExtension.

Resolution

Yellow Team: Resolved.

L-04 | Ownership Can Be Renounced And Cause Permanent DoS

Category	Severity	Location	Status
Configuration	● Low	Treasury.sol	Resolved

Description

The treasury owner is the deployer initially, and ownership will be transferred to the Timelock through a two-step transfer, after which the Timelock will accept the ownership.

However, the `renounceOwnership` function from `Ownable.sol` is not overridden and could cause a complete DoS if used by mistake during this process. Calling `renounceOwnership` not only removes the current owner but also clears the `pendingOwner`. As a result, the Timelock would not be able to accept ownership, and all funds would become permanently locked in the treasury.

Additionally, the Timelock itself can also renounce ownership via malicious governance, which would prevent the `withdraw` function in the treasury from being used.

Another point to note is that the deployer remains the owner until the Timelock accepts ownership. During this period, the deployer can withdraw funds from the treasury without any governance approval.

Recommendation

Since the treasury must always have an owner, override the `renounceOwnership` function to always revert, preventing the treasury from being left without an owner.

Resolution

Yellow Team: Resolved.

I-01 | Rebasing Tokens Cannot Be Used In Locker

Category	Severity	Location	Status
Warning	● Low	Locker.sol	Resolved

Description

The Treasury and Locker contracts determine the exact transferred amount by comparing the before and after balances of users (or the locker itself) during transfers. This enables support for fee-on-transfer tokens.

However, the Locker is not compatible with rebasing or deflationary tokens. If the underlying token balance gradually decreases over time, withdrawals will fail, particularly for later or final users. This happens because the Locker’s balance decays over time, while the withdraw function attempts to transfer the original _balances recorded at the time of locking.

Recommendation

Be aware of this limitation. No changes are necessary if rebasing tokens are not intended to be used as the underlying asset in the Locker.

Resolution

Yellow Team: Resolved.

I-02 | Mismatch Between Tests And Code Comment

Category	Severity	Location	Status
Informational	● Low	Locker.sol	Resolved

Description

The NatSpec comment in Locker states: “Single-asset vault with a 7-day time-locked withdrawal mechanism.”

However, in the tests, the immutable UNLOCK_PERIOD is set to 14 days.

Recommendation

While the tests do not necessarily need to be changed, ensure that the contract is deployed with a 7-day unlock period, or update the comment in Locker.sol accordingly.

Resolution

Yellow Team: Resolved.

I-03 | Proposals Uncancellable Once Past Pending State

Category	Severity	Location	Status
Best Practices	● Low	Locker.sol	Resolved

Description

OZ Governor v5 base `_validateCancel` (Governor.sol:787-789) only allows cancelling during Pending. Once voting starts, proposal cancellation reverts for everyone, including the proposer. This is the default behavior in the OpenZeppelin Governor implementation. Neither GovernorTimelockControl nor YellowGovernor overrides this behavior. Once voting begins, only token holders can determine the outcome and no kill switch exists.

As a result, if a malicious proposal somehow gets voted through and queued:

1. Attacker proposes something like `treasury.withdraw(token, attacker, fullBalance)`.
2. It passes the vote (social engineering, bribery, voter apathy, etc).
3. Gets queued in the timelock.
4. Community notices during the 2-day window.
5. `governor.cancel()` — reverts.
6. `timelock.cancel()` — reverts.
7. Only option is a counter-proposal, but that needs `votingDelay + votingPeriod + timelockDelay` (~10 days). The malicious one executes in day 2.
8. Anyone calls `execute()` (open EXECUTOR_ROLE), treasury gone.

While this is essentially a governance decision rather than an exploitable vulnerability, OpenZeppelin provides GovernorProposalGuardian for such edge cases, allowing proposals to be cancelled after voting when necessary.

Recommendation

Consider using OpenZeppelin’s GovernorProposalGuardian and assigning a trusted guardian (e.g., a multisig) that can cancel proposals at any stage.

However, this introduces an element of centralization. No change is required if the Yellow protocol intentionally chooses token holders/governance as the sole decision-maker without a kill switch.

Resolution

Yellow Team: Resolved.

I-04 | Misleading Error Name In Constructor Validation

Category	Severity	Location	Status
Best Practices	● Low	Locker.sol	Resolved

Description

In Locker.sol:34, when unlockPeriod_ is zero the constructor reverts with InvalidAmount(). But the unlock period is a time duration, not a token amount. The same InvalidAmount() error is also used in lock() for 0 amount deposits.

Recommendation

Consider using a specific error like InvalidPeriod().

Resolution

Yellow Team: Resolved.

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

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian 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>