AstraSec

StellaSwap stGLMR Security Audit Report

September 16, 2025

Contents

1 Introduction

1.1 About stGLMR

stGLMR by StellaSwap is a liquid staking protocol that issues a derivative token, stGLMR, for GLMR (Moonbeam's native token). The protocol enables users to stake GLMR while retaining liquidity, with stGLMR representing their staked position and accruing staking rewards over time.

The stGLMR protocol employs an interest-bearing token model, akin to Yearn vaults. Users deposit GLMR and receive stGLMR tokens, whose balance grows as staking rewards are accumulated. The protocol maintains multiple ledgers to delegate stakes to various collator candidates on Moonbeam, optimizing rewards while managing risk.

1.2 Source Code

The following source code was reviewed during the audit:

- https://github.com/stellaswap/stglmr-core
- CommitID: 05890eed94d18159434d546d9a749b4c90f68b1a

And this is the final version representing all fixes implemented for the issues identified in the audit:

- https://github.com/stellaswap/stglmr-core
- CommitID: a087bad5560798ddaeef47d87db031d01b04cfff

1.3 Revision History

Version	Date	Description
v1.0	September 16, 2025	Initial Audit

2 Overall Assessment

This report has been compiled to identify issues and vulnerabilities within the stGLMR protocol. Throughout this audit, we identified a total of 4 issues spanning various severity levels. By employing auxiliary tool techniques to supplement our thorough manual code review, we have discovered the following findings.

Severity	Count	Acknowledged	Won't Do	Addressed
Critical	–	–	–	–
High	–	–	–	–
Medium	3	1	–	2
Low	1	–	–	1
Informational	–	–	–	–
Total	4	1	–	3

3 Vulnerability Summary

3.1 Overview

Click on an issue to jump to it, or scroll down to see them all.

3.2 Security Level Reference

In web3 smart contract audits, vulnerabilities are typically classified into different severity levels based on the potential impact they can have on the security and functionality of the contract. Here are the definitions for critical-severity, high-severity, medium-severity, and low-severity vulnerabilities:

Severity	Acknowledged
C-X (Critical)	A severe security flaw with immediate and significant negative consequences. It poses high risks, such as unauthorized access, financial losses, or complete disruption of functionality. Requires immediate attention and remediation.
H-X (High)	Significant security issues that can lead to substantial risks. Although not as severe as critical vulnerabilities, they can still result in unauthorized access, manipulation of contract state, or financial losses. Prompt remediation is necessary.
M-X (Medium)	Moderately impactful security weaknesses that require attention and re-mediation. They may lead to limited unauthorized access, minor financial losses, or potential disruptions to functionality.
L-X (Low)	Minor security issues with limited impact. While they may not pose significant risks, it is still recommended to address them to maintain a robust and secure smart contract.
I-X (Informational)	Warnings and things to keep in mind when operating the protocol. No immediate action required.
U-X (Undetermined)	Identified security flaw requiring further investigation. Severity and impact need to be determined. Additional assessment and analysis are necessary.

3.3 Vulnerability Details

3.3.1 [L-1] Suggested Safeguard Launch for stGLMR

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
stGLMR.sol	Business Logic	Low	Low	Addressed

The stGLMR protocol includes a minimum initial shares validation within the stGLMR::deposit() function to mitigate share price inflation. However, this safeguard can be circumvented by a malicious actor. Specifically, an attacker could deposit and redeem stGLMR tokens, reducing the remaining shares to a minimal amount (e.g., 1 wei), and subsequently donate a large amount of GLMR to the ledger. This manipulation can artificially inflate the stGLMR share price, potentially excluding small-scale users from participation and exposing subsequent depositors to financial losses.

Remediation To mitigate the risk, the project team could deploy stGLMR with a safeguard launch mechanism, wherein the initial deposit is made by the team and never redeemed. Alternatively, implement additional validation checks within the stGLMR::redeem() function to prohibit redemption if the remaining shares fall below a predefined minimum threshold, denoted as MIN_INITIAL_SHARES.

3.3.2 [M-1] Bonding Amount Below MIN_NOMINATOR_BALANCE

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
Ledger.sol	Business Logic	Medium	Medium	Addressed

The Moonbeam staking mechanism specifies a minimum delegation amount, denoted as MIN_NOMINATOR_BALANCE. In the current implementation, when the allocated stake for a ledger falls below this threshold, the protocol does not execute the delegation for that ledger. Instead, these GLMR amounts are mistakenly treated as rewards and redirected to the FundsManager. Within the transferFromLedger() function, these amounts are subsequently re-accounted into glmrAUM and bufferedDeposits, resulting in the inflation of both metrics.

stglmr-core - Ledger.sol

```
231  if (status == LedgerTypes.LedgerStatus.Nominator || status == LedgerTypes.LedgerStatus.Idle) {
232      ...
233  } else if (status == LedgerTypes.LedgerStatus.None && diffToBond >= MIN_NOMINATOR_BALANCE) {
234
235      uint256 candidateDelegationCount = STAKING_PRECOMPILE.candidateDelegationCount(CANDIDATE);
236      uint256 delegatorDelegationCount = STAKING_PRECOMPILE.delegatorDelegationCount(address(this));
237      uint256 candidateAutoCompCount = STAKING_PRECOMPILE.candidateAutoCompoundingDelegationCount(CANDIDATE);
238
239      STAKING_PRECOMPILE.delegateWithAutoCompound{ value: diffToBond }(...);
240      status = LedgerTypes.LedgerStatus.Nominator;
241      emit Bond(address(this), diffToBond);
242  }
```

Remediation Implement proper handling of ledger stake amounts below MIN_NOMINATOR_BALANCE. This could include preventing the erroneous reclassification of these amounts as rewards, or implementing a fallback mechanism to aggregate such amounts until they meet the minimum delegation threshold

3.3.3 [M-2] Reset of ledgerStake After Delegation Revoke

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
Ledger.sol	Business Logic	Medium	Low	Addressed

When the FundManager reduces the allocation for a ledger, it initiates the unbonding process. Specifically, if the allocated amount for a ledger falls below MIN_NOMINATOR_BALANCE, the protocol revokes all delegations associated with that ledger to prevent the effective delegation amount from dropping below this threshold. However, after all delegations are revoked in this scenario, the protocol fails to reset the allocated amount, i.e., ledgerStake. This oversight can lead to unintended behavior, such as the protocol attempting to re-bond delegations for the ledger.

stglmr-core - Ledger.sol

```
259  // Where we have to empty the ledger
260  if (_ledgerStake < MIN_NOMINATOR_BALANCE && status != LedgerTypes.LedgerStatus.Idle && _activeBalance > 0) {
261      // Check if we can safely unbond without exceeding the chunks limit
262      if (canSafelyUnbond()) {
263          STAKING_PRECOMPILE.scheduleRevokeDelegation(CANDIDATE);
264          unbondingChunks.push(UnbondingChunk({
265              amount: _activeBalance,
266              requestRound: uint128(STAKING_PRECOMPILE.round())
267          }));
268
269          emit Unbond(address(this), _activeBalance);
270      } else {
271          emit UnbondingSkipped(address(this), _activeBalance, unbondingChunks.length);
272      }
273  }
```

Remediation Upon revocation of all delegations for a ledger, implement a mechanism to reset the corresponding ledgerStake to zero, ensuring that ledgerStake consistently reflects the actual delegation amount for the ledger.

3.3.4 [M-3] Potential Risks Associated with Centralization

TARGET	CATEGORY	IMPACT	LIKELIHOOD	STATUS
Multiple Contracts	Security	Medium	Low	Acknowledged

The stGLMR protocol grants significant protocol-wide control to privileged roles, introducing potential risks to its decentralization. Specifically, the SUPER_ROLE holds authority to manage all other roles, while the ROLE_LEDGER_MANAGER can set the stGLMR address, propagate new "max unlocking chunks" value to all ledgers, enable or disable multi-ledger unbonding per rebalance, and manage ledgers. Additionally, the ROLE_BEACON_MANAGER can set the ledger factory address. This concentration of control in a limited number of entities undermines the protocol's decentralized design and exposes it to potential vulnerabilities.

stglmr-core - AuthManager.sol

```
62  function add(bytes32 role, address member) external override {
63      require(_find(members[msg.sender], SUPER_ROLE) != NOT_FOUND, "FORBIDDEN");
64
65      bytes32[] storage _roles = members[member];
66
67      require(_find(_roles, role) == NOT_FOUND, "ALREADY_MEMBER");
68      _roles.push(role);
69      emit AddMember(member, role);
70  }
```

stglmr-core - FundsManager.sol

```
187  function setSTGLMR(address _stglmr) external auth(Roles.ROLE_LEDGER_MANAGER) {
188      require(ST_GLMR == address(0), "FM: ST_GLMR_ALREADY_SET");
189      require(_stglmr != address(0), "FM: INCORRECT_ST_GLMR_ADDRESS");
190      ST_GLMR = _stglmr;
191  }
```

Remediation To mitigate the identified issue, it is recommended to introduce multi-sig mechanism to undertake the role of the privileged accounts. Moreover, it is advisable to implement timelocks to govern all modifications to the privileged operations.

Response By Team This issue has been acknowledged by the team.

4 Appendix

4.1 About AstraSec

AstraSec is a blockchain security company that serves to provide high-quality auditing services for blockchain-based protocols. With a team of blockchain specialists, AstraSec maintains a strong commitment to excellence and client satisfaction. The audit team members have extensive audit experience for various famous DeFi projects. AstraSec's comprehensive approach and deep blockchain understanding make it a trusted partner for the clients.

4.2 Disclaimer

The information provided in this audit report is for reference only and does not constitute any legal, financial, or investment advice. Any views, suggestions, or conclusions in the audit report are based on the limited information and conditions obtained during the audit process and may be subject to unknown risks and uncertainties. While we make every effort to ensure the accuracy and completeness of the audit report, we are not responsible for any errors or omissions in the report.

We recommend users to carefully consider the information in the audit report based on their own independent judgment and professional advice before making any decisions. We are not responsible for the consequences of the use of the audit report, including but not limited to any losses or damages resulting from reliance on the audit report.

This audit report is for reference only and should not be considered a substitute for legal documents or contracts.

4.3 Contact

Phone	+86 156 0639 2692
Email	contact@astrasec.ai
Twitter	https://x.com/AstraSecAI